Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Compress and Control

Joel Veness, Marc G. Bellemare, Marcus Hutter, Alvin Chua, Guillaume Desjardins
Google DeepMind, Australian National University
{veness,bellemare,alschua,gdesjardins}@google.com
marcus.hutter@anu.edu.au

Abstract

This paper describes a new information-theoretic policy evaluation technique for reinforcement learning. This technique converts any compression or density model into a corresponding estimate of value. Under appropriate stationarity and ergodicity conditions, we show that the use of a sufficiently powerful model gives rise to a consistent value function estimator. We also study the behavior of this technique when applied to various Atari 2600 video games, where the use of suboptimal modeling techniques is unavoidable. We consider three fundamentally different models, all too limited to perfectly model the dynamics of the system. Remarkably, we find that our technique provides sufficiently accurate value estimates for effective on-policy control. We conclude with a suggestive study highlighting the potential of our technique to scale to large problems.

1 Introduction

Within recent years, a number of information-theoretic approaches have emerged as practical alternatives to traditional machine learning algorithms. Noteworthy examples include the compression-based approaches of Frank, Chui, and Witten (2000) and Bratko et al. (2006) to classification, and Cilibrasi and Vitányi (2005) to clustering. What differentiates these techniques from more traditional machine learning approaches is that they rely on the ability to compress the raw input, rather than combining or learning features relevant to the task at hand. Thus this family of techniques has proven most successful in situations where the nature of the data makes it somewhat unwieldy to specify or learn appropriate features. This class of methods can be formally justified by appealing to various notions within algorithmic information theory, such as Kolmogorov complexity (Li and Vitányi 2008). In this paper we show how a similarly inspired approach can be applied to reinforcement learning, or more specifically, to the tasks of policy evaluation and on-policy control.

Policy evaluation refers to the task of estimating the value function associated with a given policy, for an arbitrary given environment. The performance of well-known reinforcement learning techniques such as policy iteration (Howard 1960), approximate dynamic programming (Bertsekas and Tsitsiklis 1996; Powell 2011) and actor-critic methods (Sutton and Barto 1998), for example, all crucially depend on how well policy evaluation can be performed. In this paper we introduce a model-based approach to policy evaluation, which transforms the task of estimating a value function to that of learning a particular kind of probabilistic state model.

To better put our work into context, it is worth making the distinction between two fundamentally different classes of model based reinforcement learning methods. *Simulation based* techniques involve learning some kind of forward model of the environment from which future samples can be generated. Given access to such models, planning can be performed directly using search. Noteworthy recent examples include the work of Doshi-Velez (2009), Walsh, Goschin, and Littman (2010), Veness et al. (2010), Veness et al. (2011), Asmuth and Littman (2011), Guez, Silver, and Dayan (2012), Hamilton, Fard, and Pineau (2013) and Tziortziotis, Dimitrakakis, and Blekas (2014). Although the aforementioned works demonstrate quite impressive performance on small domains possessing complicated dynamics, scaling these methods to large state or observation spaces has proven challenging. The main difficulty that arises when using learnt forward models is that the modeling errors tend to compound when reasoning over long time horizons (Talvitie 2014).

In contrast, another family of techniques, referred to in the literature as *planning as inference*, attempt to side-step the issue of needing to perform accurate simulations by reducing the planning task to one of probabilistic inference within a generative model of the system. These ideas have been recently explored in both the neuroscience (Botvinick and Toussaint 2012; Solway and Botvinick 2012) and machine learning (Attias 2003; Poupart, Lang, and Toussaint 2011) literature. The experimental results to date have been somewhat inconclusive, making it far from clear whether the transformed problem is any easier to solve in practice. Our main contribution in this paper is to show how to set up a particularly tractable form of inference problem by generalizing compression-based classification to reinforcement learning. The key novelty is to focus the modeling effort on learning the stationary distribution of a particular kind of augmented Markov chain describing the system, from which we can approximate a type of *dual representation* (Wang, Bowling, and Schuurmans 2007; Wang et al. 2008) of the value function. Using this technique, we were able to produce effective controllers on a problem domain orders of magnitude larger than what has previously been addressed with simulation based methods.

2 Background

We start with a brief overview of the parts of reinforcement learning and information theory needed to describe our work, before reviewing compression-based classification.

2.1 Markov Decision Processes

A Markov Decision Process (MDP) is a type of probabilistic model widely used within reinforcement learning (Sutton and Barto 1998; Szepesvári 2010) and control (Bertsekas and Tsitsiklis 1996). In this work, we limit our attention to finite horizon, time homogenous MDPs whose action and state spaces are finite. Formally, an MDP is a triplet $(\mathcal{S}, \mathcal{A}, \mu)$, where $\mathcal{S}$ is a finite, non-empty set of states, $\mathcal{A}$ is a finite, non-empty set of actions and μ is the transition probability kernel that assigns to each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ a probability measure $\mu(\cdot \mid s, a)$ over $\mathcal{S} \times \mathbb{R}$. $\mathcal{S}$ and $\mathcal{A}$ are known as the *state space* and *action space* respectively. The transition probability kernel gives rise to the *state transition kernel* $\mathcal{P}(s'|s, a) := \mu(\{s'\} \times \mathbb{R} \mid s, a)$, which gives the probability of transitioning from state s to state s' if action a is taken in s.

An agent's behavior is determined by a *policy*, that defines, for each state $s \in \mathcal{S}$ and time $t \in \mathbb{N}$, a probability measure over $\mathcal{A}$ denoted by $\pi_t(\cdot \mid s)$. A *stationary policy* is a policy which is independent of time, which we will denote by $\pi(\cdot \mid s)$ where appropriate. At each time t, the agent communicates an action $A_t \sim \pi_t(\cdot \mid S_{t-1})$ to the system in state $S_{t-1} \in \mathcal{S}$. The system then responds with a state-reward pair $(S_t, R_t) \sim \mu(\cdot \mid S_{t-1}, A_t)$. Here we will assume that each reward is bounded between $[r_{\min}, r_{\max}] \subset \mathbb{R}$ and that the system starts in a state s_0 and executes for an infinite number of steps. Thus the execution of the system can be described by a sequence of random variables $A_1, S_1, R_1, A_2, S_2, R_2, ...$

The finite m-horizon *return* from time t is defined as $Z_t := \sum_{i=t}^{t+m-1} R_i$. The expected m-horizon return from time t, also known as the *value function*, is denoted by $V^\pi(s_t) := \mathbb{E}[Z_{t+1} \mid S_t = s_t]$. The return space $\mathcal{Z}$ is the set of all possible returns. The *action-value function* is defined by $Q^\pi(s_t, a_{t+1}) := \mathbb{E}[Z_{t+1} \mid S_t = s_t, A_{t+1} = a_{t+1}]$. An *optimal policy*, denoted by π^*, is a policy that maximizes the expected return $\mathbb{E}[Z_{t+1} \mid S_t]$ for all t; in our setting, a state-dependent deterministic optimal policy always exists.

2.2 Compression and Sequential Prediction

We now review sequential probabilistic prediction in the context of statistical data compression. An alphabet $\mathcal{X}$ is a set of symbols. A string of data $x_1 x_2 \ldots x_n \in \mathcal{X}^n$ of length n is denoted by $x_{1:n}$. The prefix $x_{1:j}$ of $x_{1:n}$, $j \leq n$, is denoted by $x_{\leq j}$ or $x_{<j+1}$. The empty string is denoted by ϵ. The concatenation of two strings s and r is denoted by sr.

A coding distribution ρ is a sequence of probability mass functions $\rho_n : \mathcal{X}^n \to [0, 1]$, which for all $n \in \mathbb{N}$ satisfy the constraint that $\rho_n(x_{1:n}) = \sum_{y \in \mathcal{X}} \rho_{n+1}(x_{1:n} y)$ for all $x_{1:n} \in \mathcal{X}^n$, with the base case $\rho_0(\epsilon) := 1$. From here onwards, whenever the meaning is clear from the argument to ρ, the subscript on ρ will be dropped. Under this definition, the conditional probability of a symbol x_n given previous data $x_{<n}$ is defined as $\rho(x_n | x_{<n}) := \rho(x_{1:n}) / \rho(x_{<n})$ provided $\rho(x_{<n}) > 0$, with the familiar chain rules $\rho(x_{1:n}) = \prod_{i=1}^{n} \rho(x_i | x_{<i})$ and $\rho(x_{j:k} \mid x_{<j}) = \prod_{i=j}^{k} \rho(x_i | x_{<i})$ now following.

A binary source code $c : \mathcal{X}^* \to \{0, 1\}^*$ assigns to each possible data sequence $x_{1:n}$ a binary codeword $c(x_{1:n})$ of length $\ell_c(x_{1:n})$. The typical goal when constructing a source code is to minimize the lengths of each codeword while ensuring that the original data sequence $x_{1:n}$ is always recoverable from $c(x_{1:n})$. A fundamental technique known as *arithmetic encoding* (Witten, Neal, and Cleary 1987) makes explicit the connection between coding distributions and source codes. Given a coding distribution ρ and a data sequence $x_{1:n}$, arithmetic encoding constructs a code a_ρ which produces a binary codeword whose length is essentially $-\log_2 \rho(x_{1:n})$. We refer the reader to the standard text of Cover and Thomas (1991) for further information.

2.3 Compression-based classification

Compression-based classification was introduced by Frank, Chui, and Witten (2000). Given a sequence of n labeled i.i.d. training examples $\mathcal{D} := (y_1, c_1), \ldots, (y_n, c_n)$, where y_i and c_i are the input and class labels respectively, one can apply Bayes rule to express the probability of a new example Y being classified as class $C \in \mathcal{C}$ given the training examples $\mathcal{D}$ by

$$\mathbb{P}[\, C \mid Y, \mathcal{D} \,] = \frac{\mathbb{P}[\, Y \mid C, \mathcal{D} \,] \; \mathbb{P}[\, C \mid \mathcal{D} \,]}{\sum\limits_{c \in \mathcal{C}} \mathbb{P}[\, Y \mid c, \mathcal{D} \,] \; \mathbb{P}[\, c \mid \mathcal{D} \,]}. \qquad (1)$$

The main idea behind compression-based classification is to model $\mathbb{P}[\, Y \mid C, \mathcal{D} \,]$ using a coding distribution for the inputs that is trained on the subset of examples from $\mathcal{D}$ that match class C. Well known non-probabilistic compression methods such as LEMPEL-ZIV (Ziv and Lempel 1977) can be used by forming their associated coding distribution $2^{-\ell_z(x_{1:n})}$, where $\ell_z(x_{1:n})$ is the length of the compressed data $x_{1:n}$ in bits under compression method z. The class probability $\mathbb{P}[\, C \mid \mathcal{D}]$ can be straightforwardly estimated from its empirical frequency or smoothed versions thereof. Thus the overall accuracy of the classifier essentially depends upon how well the inputs can be modeled by the class conditional coding distribution.

Compression-based classification has both advantages and disadvantages. On one hand, it is straightforward to apply generic compression techniques (including those operating at the bit or character level) to complicated input types such as richly formatted text or DNA strings (Frank, Chui, and Witten 2000; Bratko et al. 2006). On the other hand, learning a probabilistic model of the input may be significantly more difficult than directly applying standard discriminative classification techniques. Our approach to policy evaluation, which we now describe, raises similar questions.

3 Compression and Control

We now introduce *Compress and Control* (CNC), our new method for policy evaluation.

3.1 Overview

Policy evaluation is concerned with the estimation of the state-action value function $Q^\pi(s,a)$. Here we assume that the environment is a finite, time homogenous MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mu)$, and that the policy to be evaluated is a stationary Markov policy π. To simplify the exposition, we consider the finite m-horizon case, and assume that all rewards are drawn from a finite set $\mathcal{R} \subset \mathbb{R}$; later we will discuss how to remove these restrictions.

At a high level, CNC performs policy evaluation by learning a *time-independent* state-action conditional distribution $\mathbb{P}(Z \mid S, A)$; the main technical component of our work involves establishing that this time-independent conditional probability is well defined. Our technique involves constructing a particular kind of *augmented* Markov chain whose stationary distribution allows for the recovery of $\mathbb{P}(Z \mid S, A)$. Given this distribution, we can obtain

$$Q^\pi(s,a) = \sum_{z \in \mathcal{Z}} z \, \mathbb{P}(Z = z \mid S = s, A = a).$$

In the spirit of compression-based classification, CNC estimates this distribution by using Bayes rule to combine learnt density models of both $\mathbb{P}(S \mid Z, A)$ and $\mathbb{P}(Z \mid A)$. Although it might seem initially strange to learn a model that conditions on the future return, the next section shows how this counterintuitive idea can be made rigorous.

3.2 Transformation

Our goal is to define a transformed chain whose stationary distribution can be marginalized to obtain a distribution over states, actions and the m-horizon return. We need two lemmas for this purpose. To make these statements precise, we will use some standard terminology from the Markov chain literature; for more detail, we recommend the textbook of Brémaud (1999).

Definition 1. *A Homogenous Markov Chain (HMC) given by $\{X_t\}_{t\geq 0}$ over state space $\mathcal{X}$ is said to be: (AP) aperiodic iff* $\gcd\{n \geq 1 : \mathbb{P}[X_n = x | X_0 = x] > 0\} = 1, \forall x \in \mathcal{X}$*; (PR) positive recurrent iff* $\mathbb{E}[\min\{n \geq 1 : X_n = x\} | X_0 = x] < \infty, \forall x \in \mathcal{X}$*; (IR) irreducible iff* $\forall x, x' \, \exists n \geq 1 : \mathbb{P}[X_n = x' | X_0 = x] > 0$*; (EA) essentially aperiodic iff* $\gcd\{n \geq 1 : \mathbb{P}[X_n = x | X_0 = x] > 0\} \in \{1, \infty\}, \forall x \in \mathcal{X}$*. Note also that EA+IR implies AP.*

Although the term *ergodic* is sometimes used to describe particular combinations of these properties (e.g. AP+PR+IR), here we avoid it in favor of being more explicit.

Lemma 1. *Consider a stochastic process $\{X_t\}_{t\geq 1}$ over state space $\mathcal{X}$ that is independent of a sequence of $\mathcal{U}$-valued random variables $\{U_t\}_{t\geq 1}$ in the sense that $\mathbb{P}(x_t | x_{<t}, u_{<t}) = \mathbb{P}(x_t | x_{<t})$, and with U_t only depending on X_{t-1} and X_t in the sense that $\mathbb{P}(u_t | x_{1:t}, u_{<t}) = \mathbb{P}(u_t | x_{t-1}, x_t)$ and $\mathbb{P}(U_t = u | X_{t-1} = x, X_t = x')$ being independent of t. Then, if $\{X_t\}_{t\geq 1}$ is an (IR/EA/PR) HMC over $\mathcal{X}$, then $\{Y_t\}_{t\geq 1} := \{(X_t, U_t)\}_{t\geq 1}$ is an (IR/EA/PR) HMC over $\mathcal{Y} := \{y_t \in \mathcal{X} \times \mathcal{U} : \exists x_{t-1} \in \mathcal{X} : \mathbb{P}(y_t | x_{t-1}) > 0\}$.*

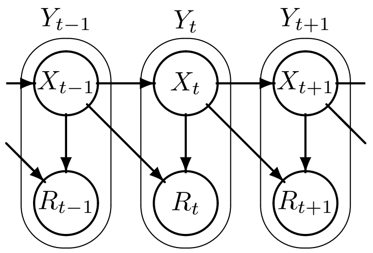

Figure 1: Lemma 1 applied to $\{((A_t, S_t), R_t)\}_{t\geq 1}$.

Lemma 1 allows HMC $\{X_t := (A_t, S_t)\}_{t\geq 1}$ to be augmented to obtain the HMC $\{Y_t := (X_t, R_t)\}_{t\geq 1}$, where A_t, S_t and R_t denote the action, state and reward at time t respectively; see Figure 1 for a graphical depiction of the dependence structure.

The second result allows the HMC $\{Y_t\}_{t\geq 1}$ to be further augmented to give the snake HMC $\{Y_{t:t+m}\}_{t\geq 1}$ (Brémaud 1999). This construction ensures that there is sufficient information within each augmented state to be able to condition on the m-horizon return.

Lemma 2. *If $\{Y_t\}_{t\geq 1}$ is an (IR/EA/PR) HMC over state space $\mathcal{Y}$, then for any $m \in \mathbb{N}$, the stochastic process $\{W_t\}_{t\geq 1}$, where $W_t := (Y_t, ..., Y_{t+m})$, is an (IR/EA/PR) HMC over $\mathcal{W} := \{y_{0:m} \in \mathcal{Y}^{m+1} : \mathbb{P}(y_{1:m} | y_0) > 0\}$.*

Now if we assume that the HMC defined by $\mathcal{M}$ and π is (IR+EA+PR), Lemmas 1 and 2 imply that there exists a unique stationary distribution ν' over the augmented state space $(\mathcal{A} \times \mathcal{S} \times \mathcal{R})^{m+1}$.

Furthermore, if we let $(A'_0, S'_0, R'_0, \ldots, A'_m, S'_m, R'_m) \sim \nu'$ and define $Z' := \sum_{i=1}^m R'_i$, it is clear that there exists a joint distribution ν over $\mathcal{Z} \times (\mathcal{A} \times \mathcal{S} \times \mathcal{R})^{m+1}$ such that $(Z', A'_0, S'_0, R'_0, \ldots, A'_m, S'_m, R'_m) \sim \nu$. Hence the ν-probability $\mathbb{P}[Z' \mid S'_0, A'_1]$ is well defined, which allows us to express the action-value function Q^π as

$$Q^\pi(s,a) = \mathbb{E}_\nu [Z' \mid S'_0 = s, A'_1 = a]. \tag{2}$$

Finally, by expanding the expectation and applying Bayes rule, Equation 2 can be further re-written as

$$\begin{aligned} Q^\pi(s,a) &= \sum_{z \in \mathcal{Z}} z \, \nu(z \mid s, a) \\ &= \sum_{z \in \mathcal{Z}} z \, \frac{\nu(s \mid z, a) \, \nu(z \mid a)}{\sum\limits_{z' \in \mathcal{Z}} \nu(s \mid z', a) \, \nu(z' \mid a)}. \end{aligned} \tag{3}$$

The CNC approach to policy evaluation involves directly learning the conditional distributions $\nu(s \mid z, a)$ and $\nu(z \mid a)$ in Equation 3 from data, and then using these learnt distributions to form a plug-in estimate of $Q^\pi(s,a)$. Notice that

$\nu(s|z,a)$ conditions on the return, similar in spirit to prior work on planning as inference (Attias 2003; Botvinick and Toussaint 2012; Solway and Botvinick 2012). The distinguishing property of CNC is that the conditioning is performed with respect to a stationary distribution that has been explicitly constructed to allow for efficient modeling and inference.

3.3 Online Policy Evaluation

We now provide an online algorithm for compression-based policy evaluation. This will produce, for all times $t \in \mathbb{N}$, an estimate $\hat{Q}^\pi_t(s,a)$ of the m-horizon expected return $Q^\pi(s,a)$ as a function of the first $t-m$ action-observation-reward triples.

Constructing our estimate involves modeling the ν-probability terms in Equation 3 using two different coding distributions, $\rho_{\rm S}$ and $\rho_{\rm Z}$ respectively; $\rho_{\rm S}$ will encode states conditional on return-action pairs, and $\rho_{\rm Z}$ will encode returns conditional on actions. Sample states, actions and returns can be generated by directly executing the system $(\mathcal{M},\pi)$; Provided the HMC $\mathcal{M}+\pi$ is (IR+EA+PR), Lemmas 1 and 2 ensure that the empirical distributions formed from a sufficiently large sample of action/state/return triples will be arbitrarily close to the required conditional ν-probabilities.

Next we describe how the coding distributions are trained. Given a history $h_t := s_0, a_1, s_1, r_1 \ldots, a_{n+m}, s_{n+m}, r_{n+m}$ with $t = n+m$, we define the m-lagged return at any time $i \leq n+1$ by $z_i := r_i + \cdots + r_{i+m-1}$. The sequence of the first n states occurring in h_t can be mapped to a subsequence denoted by $s^{z,a}_{0:n-1}$ that is defined by keeping only the states $(s_i : z_{i+1} = z \wedge a_{i+1} = a)_{i=0}^{n-1}$. Similarly, a sequence of m-lagged returns $z_{1:n}$ can be mapped to a subsequence $z^a_{1:n}$ formed by keeping only the returns $(z_i : a_i = a)_{i=1}^n$ from $z_{1:n}$. Our value estimate at time t of taking action a in state s can now be defined as

$$\hat{Q}^\pi_t(s,a) := \sum_{z\in\mathcal{Z}} z\, w^{z,a}_t(s), \tag{4}$$

where

$$w^{z,a}_t(s) := \frac{\rho_{\rm S}(\, s \mid s^{z,a}_{0:n-1}\,)\, \rho_{\rm Z}(z \mid z^a_{1:n})}{\sum\limits_{z'\in\mathcal{Z}} \rho_{\rm S}(s \mid s^{z',a}_{0:n-1})\, \rho_{\rm Z}(z' \mid z^a_{1:n})} \tag{5}$$

approximates the probability of receiving a return of z if action a is selected in state s.

Implementation. The action-value function estimate $\hat{Q}^\pi_t$ can be computed efficiently by maintaining $|\mathcal{Z}||\mathcal{A}|$ *buckets*, each corresponding to a particular return-action pair (z,a). Each bucket contains an instance of the coding distribution $\rho_{\rm S}$ encoding the state sequence $s^{z,a}_{0:n-1}$. Similarly, $|\mathcal{A}|$ buckets containing instances of $\rho_{\rm Z}$ are created to encode the various return subsequences. This procedure is summarized in Algorithm 1.

To obtain a particular state-action value estimate, Equations 4 and 5 can be computed directly by querying the appropriate bucketed coding distributions. Assuming that the time required to compute each conditional probability using $\rho_{\rm S}$ and $\rho_{\rm Z}$ is constant, the time complexity for computing $\hat{Q}_t(s,a)$ is $O(|\mathcal{Z}|)$.

Algorithm 1 CNC POLICY EVALUATION

```
Require: Stationary policy $\pi$, environment $\mathcal{M}$
Require: Finite planning horizon $m \in \mathbb{N}$
Require: Coding distributions $\rho_{\rm S}$ and $\rho_{\rm Z}$
1: for $i = 1$ to $t$ do
2:     Perform $a_i \sim \pi(\cdot \mid s_{i-1})$
3:     Observe $(s_i, r_i) \sim \mu(\cdot \mid s_{i-1}, a_i)$
4:     if $i \geq m$ then
5:         Update $\rho_{\rm S}$ in bucket $(z_{i-m+1}, a_{i-m+1})$ with $s_{i-m}$
6:         Update $\rho_{\rm Z}$ in bucket $a_{i-m+1}$ with $z_{i-m+1}$
7:     end if
8: end for
9: return $\hat{Q}^\pi_t$
```

3.4 Analysis

We now show that the state-action estimates defined by Equation 4 are consistent provided that consistent density estimators are used for both $\rho_{\rm S}$ and $\rho_{\rm Z}$. Also, we will say f_n converges stochastically to 0 with rate $n^{-1/2}$ if and only if

$$\exists c > 0,\ \forall \delta \in [0,1] : \mathbb{P}\Big(|f_n(\omega)| \leq \sqrt{\tfrac{c}{n}\ln\tfrac{2}{\delta}}\,\Big) \geq 1-\delta,$$

and will denote this by writing $f_n(\omega) \in O_{\mathbb{P}}(n^{-1/2})$.

Theorem 1. *Given an m-horizon, finite state space, finite action space, time homogenous MDP $\mathcal{M} := (\mathcal{S},\mathcal{A},\mu)$ and a stationary policy π that gives rise to an (IR+EA+PR) HMC, for all $\epsilon > 0$, we have that for any state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ that*

$$\lim_{n\to\infty} \mathbb{P}\left[\, |\, \hat{Q}^\pi_n(s,a) - Q^\pi(s,a)\, |\ \geq \epsilon \right] = 0,$$

provided $\rho_{\rm S}$ and $\rho_{\rm Z}$ are consistent estimators of $\nu(s|z,a)$ and $\nu(z|a)$ respectively. Furthermore, if $|\rho_{\rm S}(s|z,a) - \nu(s|z,a)| \in O_{\mathbb{P}}(n^{-1/2})$ and $|\rho_{\rm Z}(z|a) - \nu(z|a)| \in O_{\mathbb{P}}(n^{-1/2})$ then $|\hat{Q}^\pi_n(s,a) - Q^\pi(s,a)| \in O_{\mathbb{P}}(n^{-1/2})$.

Next we state consistency results for two types of estimators often used in model-based reinforcement learning.

Theorem 2. *The frequency estimator $\rho(x_n|x_{<n}) := \frac{1}{n-1}\sum_{i=1}^{n-1} [\![x_n = x_i]\!]$ when used as either $\rho_{\rm S}$ or $\rho_{\rm Z}$ is a consistent estimator of $\nu(s|z,a)$ or $\nu(z|a)$ respectively for any $s \in \mathcal{S}$, $z \in \mathcal{Z}$, and $a \in \mathcal{A}$; furthermore, the absolute estimation error converges stochastically to 0 with rate $n^{-1/2}$.*

Note that the above result is essentially tabular, in the sense that each state is treated atomically. The next result applies to a factored application of multi-alphabet Context Tree Weighting (CTW) (Tjalkens, Shtarkov, and Willems 1993; Willems, Shtarkov, and Tjalkens 1995; Veness et al. 2011), which can handle considerably larger state spaces in practice. In the following, we use the notation $s_{n,i}$ to refer to the ith factor of state s_n.

Theorem 3. *Given a state space that is factored in the sense that $\mathcal{S} := \mathcal{B}_1 \times \cdots \times \mathcal{B}_k$, the estimator $\rho(s_n \mid s_{<n}) := \prod_{i=1}^{k} \mathrm{CTW}(s_{n,i} \mid s_{n,<i}, s_{<n,1:i})$ when used as ρ_S, is a consistent estimator of $\nu(s|z,a)$ for any $s \in \mathcal{S}$, $z \in \mathcal{Z}$, and $a \in \mathcal{A}$; furthermore, the absolute estimation error converges stochastically to 0 at a rate of $n^{-1/2}$.*

4 Experimental Results

In this section we describe two sets of experiments. The first set is an experimental validation of our theoretical results using a standard policy evaluation benchmark. The second combines CNC with a variety of density estimators and studies the resulting behavior in a large on-policy control task.

4.1 Policy Evaluation

Our first experiment involves a simplified version of the game of Blackjack (Sutton and Barto 1998, Section 5.1). In Blackjack, the agent requests cards from the dealer. A game is won when the agent's card total exceeds the dealer's own total. We used CNC to estimate the value of the policy that stays if the player's sum is 20 or 21, and hits in all other cases. A state is represented by the single card held by the dealer, the player's card total so far, and whether the player holds a usable ace. In total, there are 200 states, two possible actions (hit or stay), and three possible returns (-1, 0 and 1). A Dirichlet-Multinomial model with hyper-parameters $\alpha_i = \frac{1}{2}$ was used for both ρ_S and ρ_Z.

Figure 2 depicts the estimated MSE and average maximum squared error of $\hat{Q}^{\pi}$ over 100,000 episodes; the mean and maximum are taken over all possible state-action pairs and averaged over 10,000 trials. We also compared CNC to a first-visit Monte Carlo value estimate (Szepesvári 2010). The CNC estimate closely tracks the Monte Carlo estimate, even performing slightly better early on due to the smoothing introduced by the use of a Dirichlet prior. As predicted by the analysis in Section 3.4, the MSE decays toward zero.

4.2 On-policy Control

Our next set of experiments explored the on-policy control behavior of CNC under an ϵ-greedy policy. The purpose of these experiments is to demonstrate the potential of CNC to scale to large control tasks when combined with a variety of different density estimators. Note that Theorem 1 does not apply here: using CNC in this way violates the assumption that π is stationary.

Evaluation Platform. We evaluated CNC using ALE, the Arcade Learning Environment (Bellemare et al. 2013), a reinforcement learning interface to the Atari 2600 video game platform. Observations in ALE consist of frames of 160×210 7-bit color pixels generated by the Stella Atari 2600 emulator. Although the emulator generates frames at 60Hz, in our experiments we consider time steps that last 4 consecutive frames, following the existing literature (Bellemare, Veness, and Talvitie 2014; Mnih et al. 2013). We first focused on the game of PONG, which has an action space of $\{\text{UP}, \text{DOWN}, \text{NOOP}\}$ and provides a reward of 1 or -1 whenever a point is scored by either the agent or its computer opponent. Episodes end when either player has scored

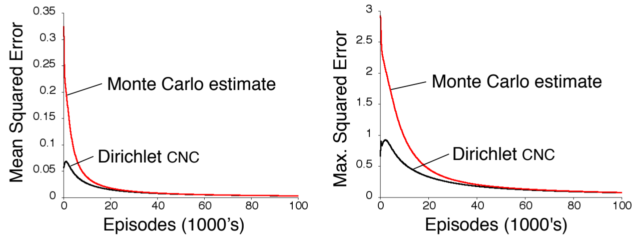

Figure 2: Mean and maximum squared errors of the Monte Carlo and CNC estimates on the game of Blackjack.

21 points; as a result, possible scores for one episode range between -21 to 21, with a positive score corresponding to a win for the agent.

Experimental Setup. We studied four different CNC agents, with each agent corresponding to a different choice of model for ρ_S; the Sparse Adapative Dirichlet (SAD) estimator (Hutter 2013) was used for ρ_Z for all agents. Each agent used an ϵ-greedy policy (Sutton and Barto 1998) with respect to its current value function estimates. The exploration rate ϵ was initialized to 1.0, then decayed linearly to 0.02 over the course of 200,000 time steps. The horizon was set to $m = 80$ steps, corresponding to roughly 5 seconds of play. The agents were evaluated over 10 trials, each lasting 2 million steps.

The first model we consider is a factored application of the SAD estimator, a count based model designed for large, sparse alphabets. The model divides the screen into 16×16 regions. The probability of a particular image patch occurring within each region is modeled using a region-specific SAD estimator. The probability assigned to a whole screen is the product of the probabilities assigned to each patch.

The second model is an auto-regressive application of logistic regression (Bishop 2006), that assigns a probability to each pixel using a shared set of parameters. The product of these per-pixel probabilities determines the probability of a screen under this model. The features for each pixel prediction correspond to the pixel's local context, similar to standard context-based image compression techniques (Witten, Moffat, and Bell 1999). The model's parameters were updated online using ADAGRAD (Duchi, Hazan, and Singer 2011). The hyperparameters (including learning rate, choice of context, etc.) were optimized via the random sampling technique of Bergstra and Bengio (2012).

The third model uses the LEMPEL-ZIV algorithm (Ziv and Lempel 1977), a dictionary-based compression technique. It works by adapting its internal data structures over time to assign shorter code lengths to more frequently seen substrings of data. For our application, the pixels in each frame were encoded in row-major order, by first searching for the longest sequence in the history matching the new data to be compressed, and then encoding a triple that describes the temporal location of the longest match, its length, as well as the next unmatched symbol. This process repeats until no data is left. Recalling Section 2.3, the (implicit) conditional probability of a state s under the LEMPEL-ZIV model can

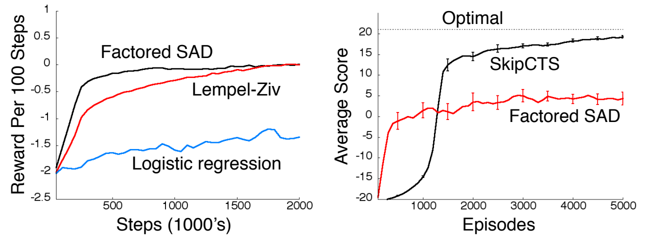

Figure 3: *Left.* Average reward over time in PONG. *Right.* Average score across episodes in PONG. Error bars indicate one inter-trial standard error.

now be obtained by computing

$$\rho_S(s \mid s^{z,a}_{0:n-1}) := 2^{-\left[\ell_{LZ}(s^{z,a}_{0:n-1}s) - \ell_{LZ}(s^{z,a}_{0:n-1})\right]}.$$

Results. As depicted in Figure 3 (left), all three models improved their policies over time. By the end of training, two of these models had learnt control policies achieving win rates of approximately 50% in PONG. Over their last 50 episodes of training, the LEMPEL-ZIV agents averaged -0.09 points per episode (std. error: 1.79) and the factored SAD agents, 3.29 (std. error: 2.49). While the logistic regression agents were less successful (average -17.87, std. error 0.38) we suspect that further training time would significantly improve their performance. Furthermore, all agents ran at real-time or better. These results highlight how CNC can be successfully combined with fundamentally different approaches to density estimation.

We performed one more experiment to illustrate the effects of combining CNC with a more sophisticated density model. We used SKIPCTS, a recent Context Tree Weighting derivative, with a context function tailored to the ALE observation space (Bellemare, Veness, and Talvitie 2014). As shown in Figure 3 (right), CNC combined with SKIPCTS learns a near-optimal policy in PONG. We also compared our method to existing results from the literature (Bellemare et al. 2013; Mnih et al. 2013), although note that the DQN scores, which correspond to a different training regime and do not include Freeway, are included only for illustrative purposes. As shown in Figure 4, CNC can also learn competitive control policies on FREEWAY and Q*BERT.

Interestingly, we found SKIPCTS to be insufficiently accurate for effective MCTS planning when used as a forward model, even with enhancements such as double progressive widening (Couëtoux et al. 2011). In particular, our best simulation-based agent did not achieve a score above -14 in PONG, and performed no better than random in Q*BERT and FREEWAY. In comparison, our CNC variants performed significantly better using orders of magnitude less computation. While it would be premature to draw any general conclusions, the CNC approach does appear to be more forgiving of modeling inaccuracies.

5 Discussion and Limitations

The main strength and key limitation of the CNC approach seems to be its reliance on an appropriate choice of density estimator. One could only expect the method to perform well if the learnt models can capture the observational structure specific to high and low return states. Specifying a model can be thus viewed as committing to a particular kind of compression-based similarity metric over the state space. The attractive part of this approach is that density modeling is a well studied area, which opens up the possibility of bringing in many ideas from machine learning, statistics and information theory to address fundamental questions in reinforcement learning. The downside of course is that density modeling is itself a difficult problem. Further investigation is required to better understand the circumstances under which one would prefer CNC over more traditional model-free approaches that rely on function approximation to scale to large and complex problems.

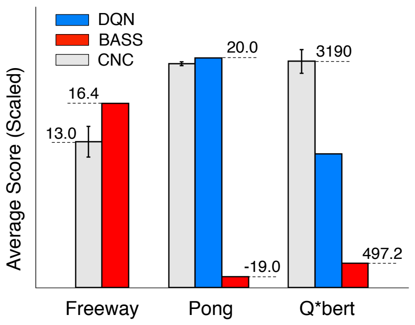

Figure 4: Average score over the last 500 episodes for three Atari 2600 games. Error bars indicate one inter-trial one standard error.

So far we have only applied CNC to undiscounted, finite horizon problems with finite action spaces, and more importantly, finite (and rather small) return spaces. This setting is favorable for CNC, since the per-step running time depends on $|\mathcal{Z}| \leq m|r_{max} - r_{min}|$; in other words, the worst case running time scales no worse than linearly in the length of the horizon. However, even modest changes to the above setting can change the situation drastically. For example, using discounted return can introduce an exponential dependence on the horizon. Thus an important topic for future work is to further develop the CNC approach for large or continuous return spaces. Since the return space is only one dimensional, it would be natural to consider various discretizations of the return space. For example, one could consider a tree based discretization that recursively subdivides the return space into successively smaller halves. A binary tree of depth d would produce 2^d intervals of even size with an accuracy of $\epsilon = m(r_{max} - r_{min})/2^d$. This implies that to achieve an accuracy of at least ϵ we would need to set $d \geq \log_2(m(r_{max} - r_{min})/\epsilon)$, which should be feasible for many applications. Furthermore, one could attempt to adaptively learn the best discretization (Hutter 2005a) or approximate Equation 4 using Monte Carlo sampling. These enhancements seem necessary before we could consider applying CNC to the complete suite of ALE games.

6 Closing Remarks

This paper has introduced CNC, an information-theoretic policy evaluation and on-policy control technique for reinforcement learning. The most interesting aspect of this approach is the way in which it uses a learnt probabilistic model that conditions on the future return; remarkably, this counterintuitive idea can be justified both in theory and in practice.

While our initial results show promise, a number of open questions clearly remain. For example, so far the CNC value estimates were constructed by using only the Monte Carlo return as the learning signal. However, one of the central themes in Reinforcement Learning is *bootstrapping*, the idea of constructing value estimates on the basis of other value estimates (Sutton and Barto 1998). A natural question to explore is whether bootstrapping can be incorporated into the learning signal used by CNC.

For the case of on-policy control, it would be also interesting to investigate the use of compression techniques or density estimators that can automatically adapt to non-stationary data. A promising line of investigation might be to consider the class of meta-algorithms given by György, Linder, and Lugosi (2012), that can convert any stationary coding distribution into its piece-wise stationary extension; efficient algorithms from this class have shown promise for data compression applications, and come with strong theoretical guarantees (Veness et al. 2013). Furthermore, extending the analysis in Section 3.4 to cover the case of on-policy control or policy iteration (Howard 1960) would be highly desirable.

Finally, we remark that information-theoretic perspectives on reinforcement learning have existed for some time; in particular, Hutter (2005b) described a unification of algorithmic information theory and reinforcement learning, leading to the AIXI optimality notion for reinforcement learning agents. Establishing whether any formal connection exists between this body of work and ours is deferred to the future.

Acknowledgments. We thank Kee Siong Ng, Andras György, Shane Legg, Laurent Orseau and the anonymous reviewers for their helpful feedback on earlier revisions.

References

Asmuth, J., and Littman, M. L. 2011. Learning is planning: near Bayes-optimal reinforcement learning via Monte-Carlo tree search. In *Uncertainty in Artificial Intelligence (UAI)*, 19–26.

Attias, H. 2003. Planning by Probabilistic Inference. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*.

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 (JAIR)* 47:253–279.

Bellemare, M. G.; Veness, J.; and Talvitie, E. 2014. Skip Context Tree Switching. In *Proceedings of the Thirty-First International Conference on Machine Learning (ICML)*.

Bergstra, J., and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)* 13:281–305.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Botvinick, M., and Toussaint, M. 2012. Planning as inference. In *Trends in Cognitive Sciences 10*, 485–588.

Bratko, A.; Cormack, G. V.; R, D.; Filipi, B.; Chan, P.; Lynam, T. R.; and Lynam, T. R. 2006. Spam filtering using statistical data compression models. *Journal of Machine Learning Research (JMLR)* 7:2673–2698.

Brémaud, P. 1999. *Markov chains : Gibbs fields, Monte Carlo simulation and queues*. Texts in applied mathematics. New York, Berlin, Heidelberg: Springer.

Cilibrasi, R., and Vitányi, P. M. B. 2005. Clustering by compression. *IEEE Transactions on Information Theory* 51:1523–1545.

Couëtoux, A.; Hoock, J.-B.; Sokolovska, N.; Teytaud, O.; and Bonnard, N. 2011. Continuous upper confidence trees. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, 433–445. Springer-Verlag.

Cover, T. M., and Thomas, J. A. 1991. *Elements of information theory*. New York, NY, USA: Wiley-Interscience.

Doshi-Velez, F. 2009. The Infinite Partially Observable Markov Decision Process. In *Advances in Neural Information Processing Systems (NIPS) 22*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)* 12:2121–2159.

Frank, E.; Chui, C.; and Witten, I. H. 2000. Text categorization using compression models. In *Proceedings of Data Compression Conference (DCC)*, 200–209. IEEE Computer Society Press.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient Bayes-Adaptive Reinforcement Learning using Sample-based Search. In *Advances in Neural Information Processing Systems (NIPS) 25*.

György, A.; Linder, T.; and Lugosi, G. 2012. Efficient Tracking of Large Classes of Experts. *IEEE Transactions on Information Theory* 58(11):6709–6725.

Hamilton, W. L.; Fard, M. M.; and Pineau, J. 2013. Modelling Sparse Dynamical Systems with Compressed Predictive State Representations. In *ICML*, volume 28 of *JMLR Proceedings*, 178–186.

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

Hutter, M. 2005a. Fast non-parametric Bayesian inference on infinite trees. In *Proceedings of 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 144–151.

Hutter, M. 2005b. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer.

Hutter, M. 2013. Sparse adaptive dirichlet-multinomial-like processes. In *Conference on Computational Learning Theory (COLT)*, 432–459.

Li, M., and Vitányi, P. 2008. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, third edition.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Poupart, P.; Lang, T.; and Toussaint, M. 2011. Escaping Local Optima in POMDP Planning as Inference. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, 1263–1264.

Powell, W. B. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2nd edition.

Solway, A., and Botvinick, M. 2012. Goal-directed decision making as probabilistic inference: A computational framework and potential neural correlates. *Psychololological Review* 119:120–154.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Szepesvári, C. 2010. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Talvitie, E. 2014. Model Regularization for Stable Sample Rollouts. In *Uncertainty in Artificial Intelligence (UAI)*.

Tjalkens, T. J.; Shtarkov, Y. M.; and Willems, F. M. J. 1993. Context tree weighting: Multi-alphabet sources. In *Proceedings of the 14th Symposium on Information Theory Benelux*.

Tziortziotis, N.; Dimitrakakis, C.; and Blekas, K. 2014. Cover Tree Bayesian Reinforcement Learning. *Journal of Machine Learning Research (JMLR)* 15:2313–2335.

Veness, J.; Ng, K. S.; Hutter, M.; and Silver, D. 2010. Reinforcement Learning via AIXI Approximation. In *Proceedings of the Conference for the Association for the Advancement of Artificial Intelligence (AAAI)*.

Veness, J.; Ng, K. S.; Hutter, M.; Uther, W.; and Silver, D. 2011. A Monte Carlo AIXI approximation. *Journal of Artificial Intelligence Research (JAIR)* 40:95–142.

Veness, J.; White, M.; Bowling, M.; and Gyorgy, A. 2013. Partition Tree Weighting. In *Proceedings of Data Compression Conference (DCC)*, 321–330.

Walsh, T. J.; Goschin, S.; and Littman, M. L. 2010. Integrating Sample-Based Planning and Model-Based Reinforcement Learning. In *Proceedings of the Conference for the Association for the Advancement of Artificial Intelligence (AAAI)*.

Wang, T.; Bowling, M.; Schuurmans, D.; and Lizotte, D. J. 2008. Stable dual dynamic programming. In *Advances in Neural Information Processing Systems (NIPS) 20*, 1569–1576.

Wang, T.; Bowling, M.; and Schuurmans, D. 2007. Dual representations for dynamic programming and reinforcement learning. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 44–51.

Willems, F. M.; Shtarkov, Y. M.; and Tjalkens, T. J. 1995. The Context Tree Weighting Method: Basic Properties. *IEEE Transactions on Information Theory* 41:653–664.

Witten, I. H.; Moffat, A.; and Bell, T. C. 1999. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann.

Witten, I. H.; Neal, R. M.; and Cleary, J. G. 1987. Arithmetic coding for data compression. *Communications of the ACM*. 30:520–540.

Ziv, J., and Lempel, A. 1977. A universal algorithm for sequential data compression. *Information Theory, IEEE Transactions on* 23(3):337–343.