Policy-Gradients for PSRs and POMDPs

Douglas Aberdeen
doug.aberdeen@anu.edu.au

Olivier Buffet
olivier.buffet@nicta.com.au

Owen Thomas
owen.thomas@nicta.com.au

National ICT Australia
Locked Bag 8001, Canberra
Australia

Abstract

In uncertain and partially observable environments control policies must be a function of the complete history of actions and observations. Rather than present an ever growing history to a learner, we instead track *sufficient statistics* of the history and map those to a control policy. The mapping has typically been done using dynamic programming, requiring large amounts of memory. We present a general approach to mapping sufficient statistics directly to control policies by combining the tracking of sufficient statistics with the use of policy-gradient reinforcement learning. The best known sufficient statistic is the belief state, computed from a known or estimated partially observable Markov decision process (POMDP) model. More recently, predictive state representations (PSRs) have emerged as a potentially compact model of partially observable systems. Our experiments explore the usefulness of both of these sufficient statistics, exact and estimated, in direct policy-search.

1 Introduction

Acting in partially observable domains without a model of the system dynamics is challenging. In general, policies must map entire *histories* to actions in order to obtain the optimal policy possible under partial observability. Fortunately, it is not necessary to represent the entire history to a learner. Instead a *sufficient statistic* of the history can be used.

Sufficient statistics are those where no information relevant to predicting well is lost. A simple example is the POMDP belief state that represents the probability of being in each underlying state of the system. A control policy can then be found — using methods such as value iteration — by using sufficient statistics instead of the underlying MDP states [James *et al.*, 2004]. The drawback is that such algorithms are typically *at best* PSpace-complete with respect to the size of the state space. Even small problems may fail to converge to a sensible policy due to memory consumption.

We propose an alternative approach, combining sufficient statistics generated with true or estimated models, with direct policy-search methods from reinforcement learning. In particular, we show how any sufficient statistic can be mapped directly to control policies using policy-gradient reinforcement learning [Sutton *et al.*, 2000; Baxter *et al.*, 2001]. Even though gradient methods guarantee only local convergence, they are attractive because their memory usage is independent of the size or complexity of the state space.

The POMDP belief state is the well know sufficient statistic for partially observable dynamical systems. To demonstrate the applicability of our approach to *any* sufficient statistic we also compare using POMDP belief states to the more recent predictive state representation (PSR) of dynamical systems [Littman *et al.*, 2002; Singh *et al.*, 2004]. PSRs are interesting because they offer a potentially more compact representation than POMDPs. Also, the fact that they are based directly on observable "tests" makes them amenable to discovering the PSR model from action/observation trajectories. We believe this paper represents the first use of PSR models for direct policy-search, and the first use of *estimated* PSR models for any kind of control. For this reason, much of the paper is dedicated to describing PSR models for direct policy search.

Because sufficient statistics are usually vectors we need reinforcement learning methods suited to function approximation. Previous approaches to planning with PSRs have assumed access to the exact PSR model and hence adopted exact planning algorithms, such as incremental pruning [James *et al.*, 2004]. Reinforcement learning methods that are not known to be robust under function approximation have also been used, such

as SARSA [Rafols *et al.*, 2005], or Q-learning [James *et al.*, 2004]. These methods also fail to scale when there are many underlying states, or many PSR core-tests. Our choice of algorithm is the Natural Actor-Critic (NAC) algorithm [Peters *et al.*, 2005], which is guaranteed to converge to at least a local maximum even under function approximation and partial observability. NAC also has memory usage that is independent of the state space.

Our experiments focus on a comparison of policies learned using various statistics including: estimated and exact PSR prediction vectors and estimated and exact POMDP belief states. We found that exact (and estimated) PSRs do indeed allow better than reactive policies to be learned. However, we also found that, when mapping statistics to action distributions, POMDP belief states have the advantage that linear mappings are particularly easy to learn for the case where the belief state accurately identifies the true state. A final contribution of this paper is to note that the PSR discovery problem may in practice be equally difficult to problem of estimating the number of hidden MDP states, potentially negating a key benefit of PSRs in some cases.

2 POMDPs

Partially observable Markov Decision Processes have been popular for representing dynamic systems with uncertainty. Given the current (hidden) state of the system $s \in \mathcal{S}$, and a control action $a \in \mathcal{A}$, the probability of next state s' is given by $\Pr(s'|s,a)$. An observation $o \in \mathcal{O}$ is then generated with probability $\Pr(o|s',a)$, dependent only on the action and next state. We assume finite state, observation, and action spaces. Rewards are given for state transitions $r(a,s')$. The reward at step i is denoted $r_i := r(a_i, s_i)$. We seek to maximise the normalised *discounted* sum of rewards

$$R(\boldsymbol{\theta}) = \mathbb{E}\left\{ (1-\gamma) \sum\nolimits_{t=0}^{\infty} \gamma^t r_i \middle| \boldsymbol{\theta} \right\}.$$

We assume a parameterised stochastic control policy where the probability of choosing action a_{i+1} is given by $\Pr(a_{i+1}|h_i,\boldsymbol{\theta})$, where h is the length i history of observations and actions, $h = a_1 o_1 a_2 o_2 \ldots a_i o_i$, and $\boldsymbol{\theta}$ is matrix of real valued parameters.

It is generally not feasible for a policy to depend directly on an exponentially large number of possible histories. Instead, actions are computed from a sufficient statistic of history. A simple example is the POMDP belief state $\mathbf{b}$, a vector giving the probability of each underlying state $s \in \mathcal{S}$. Given a belief state for time index $i-1$, the next belief state is computed from the chosen action a_i, the observation o_i

$$\mathbf{b}_i(s') = \frac{\Pr(o_i|s',a_i) \sum_{s\in\mathcal{S}} \mathbf{b}_{i-1}(s) \Pr(s'|s,a_i)}{\sum_{s''\in\mathcal{S}} \Pr(o_i|s'',a_i) \sum_{s\in\mathcal{S}} \mathbf{b}_{i-1}(s) \Pr(s''|s,a_i)}. \quad (1)$$

Unfortunately this approach requires knowledge of model parameters $\Pr(s'|s,a_i)$ and $\Pr(o|s')$.

2.1 Model-Free RL in POMDPs

A non-sufficient, but simple, statistic is to use the last k observations and actions. We can easily create a table of all possible combinations of the last k (observation,action) pairs mapped to action likelihoods. But this approach is not desirable because it requires $O((|\mathcal{O}||\mathcal{A}|)^k|\mathcal{A}|)$ parameters for an exact representation and fails if more than k steps of history are required. This is an example of a k-order Markov model [Littman *et al.*, 2002]. We could consider adapting the amount of history automatically, extending the history when statistical tests indicate the inability of the system to fully resolve hidden state, e.g., the U-Tree algorithm of McCallum [1996].

Alternatively, we could *learn* the POMDP parameters and compute a sufficient statistic using (1) and the estimated model. Chrisman [1992] did this with a Baum-Welsh style EM estimation of the parameters from a long window of history, essentially a modified form of hidden Markov model training. Littman *et al.* [2002] suggest that this approach requires a good initial estimate of the model.

3 Predictive State Representations

Predictive state representations [Littman *et al.*, 2002] are an alternative to POMDPs for modelling dynamic systems with hidden state. Instead of POMDP belief states, PSRs maintain a prediction vector that gives the probability of observing each of a small set of action/observation sequences called "tests". As we shall see, the PSR equivalent of (1) is a linear transformation of the current prediction vector using a set of PSR parameters. The power of PSRs is in the proof that a finite number of test predictions is a sufficient statistic for the system, potentially allowing policies with access to the PSR prediction vector to act as optimally as possible under partial observability. Moreover, PSR vectors are potentially more compact than POMDP belief states because it is guaranteed that the number of core tests will be equal to, *or less than*, the number of underlying states [Singh *et al.*, 2004].

We now adopt the description of PSRs given by Bowling *et al.* [2006]. A test t is a sequence $t = a_1 o_1 a_2 o_2 \ldots a_i o_i$ of action/observation pairs that may occur in the future. The test succeeds if $o_1 o_2 \ldots o_i$ is

observed after performing actions $a_1 a_2 \dots a_i$. The null test ε succeeds by definition and is always part of the test set. A prediction $\Pr(t|h)$ is the probability that test t succeeds given the history and assuming the policy chooses the actions specified in the test. Suppose there exists a *core* set of tests $\mathcal{Q}$ such that the prediction for *any* other test can be computed from a linear combination of the predictions of tests in $\mathcal{Q}$. If this is the case the system can be modelled compactly as a linear PSR. More precisely, if we follow the standard PSR convention of defining $\Pr(\mathcal{Q}|h)$ as a $|\mathcal{Q}|$ row vector of predictions for the core tests, and $\mathbf{m}_t$ as a $|\mathcal{Q}|$ column vector of parameters for test t, then for all possible tests $\Pr(t|h) = \Pr(\mathcal{Q}|h)\mathbf{m}_t$. In other words, $\Pr(\mathcal{Q}|h)$ entirely summarises the history.

Given a history h, a new action a and subsequent observation o, we can update the PSR vector $\Pr(\mathcal{Q}|h)$ by updating each element of $\Pr(q|h)$ for all $q \in \mathcal{Q}$

$$\Pr(q|hao) = \frac{\Pr(haoq)}{\Pr(hao)} = \frac{\Pr(aoq|h)\Pr(h)}{\Pr(ao|h)\Pr(h)} = \frac{\Pr(aoq|h)}{\Pr(ao|h)}.$$

Because the probability that t succeeds can be predicted with $\Pr(t|h) = \Pr(\mathcal{Q}|h)\mathbf{m}_t$ we have

$$\Pr(q|hao) = \frac{\Pr(\mathcal{Q}|h)\mathbf{m}_{aoq}}{\Pr(\mathcal{Q}|h)\mathbf{m}_{ao}}. \quad (2)$$

The *learning* problem is to find the $|\mathcal{Q}| \times 1$ vectors $\mathbf{m}_t$ for all core tests $\mathcal{Q}$ and all tests that are single step prefix extensions aoq $\forall q \in \mathcal{Q}, a \in \mathcal{A}, o \in \mathcal{O}$, the union of which we denote by $\mathcal{X}$. The *discovery* problem is finding the minimum set of core tests $\mathcal{Q}$.

4 RL with Online PSRs

Our goal is to learn good control policies purely from experience. We first describe the PSR estimation algorithm. We adopt the online constrained-gradient algorithm of McCracken and Bowling [2006] because 1) it deals with both the learning and discovery problems from experience rather than from the POMDP model; 2) it does not require resets; 3) it avoids the bias problems associated with some Monte-Carlo estimators [Bowling *et al.*, 2006]; 4) it admits the possibility of learning online, simultaneously with the policy, which may reduce the total number of samples required to learn, and may in fact be helpful in large or non-stationary domains.

4.1 Online PSR Learning

Algorithm 1 is an alternative presentation of the constrained gradient algorithm for online learning and discovery of PSRs [McCracken and Bowling, 2006]. The algorithm continues until a maximum number of candidate core tests have been found. Lines 6–8 update predictions for each $t \in \mathcal{X}$. The routine `expand`($\mathcal{Q}$) adds tests to $\mathcal{T}$ to guarantee:

1. if $q \in \mathcal{Q}$ then $a_i o_j q \in \mathcal{T}$ $\forall a_i \in \mathcal{A}$ and $o_j \in \mathcal{O}$;
2. if $tao \in \mathcal{T}$ then $t \in \mathcal{T}$;
3. if $tao \in \mathcal{T}$ then $tao_j \in \mathcal{T}$ $\forall o_j \in \mathcal{O}$.

Because of these conditions lines 6–8 are guaranteed to update all the core tests, producing a PSR vector for step i. Lines 9–12 use the updated PSR vector to estimate the value of all other tests by solving a series of least squares problems for $\mathbf{m}_t$. The updated prediction is then $\Pr(t|h) = \Pr(\mathcal{Q}|h)\mathbf{m}_t$. Least squares estimation can introduce errors (such as values outside [0,1]). In line 13 the routine `normalise`($\Pr(\mathcal{T}|h_i)$) constrains all entries to obey

$$\Pr(tao_j|h_i) \leftarrow \frac{\Pr(t|h_i)\Pr(tao_j|h_i)}{\sum_{o' \in \mathcal{O}} \Pr(tao'|h_i)} \quad \forall o_j \in \mathcal{O},$$

which restricts the probability of all one step extensions of test t to sum to $\Pr(t|h_i)$.

So far we have only achieved the update of all the test predictions. Lines 14–17 find tests that are about to succeed and performs a TD like gradient step to make the prediction value of a successful test $\Pr(tao|h_i)$ closer to that of its parent's value $\Pr(t|h_i)$. This tunes the PSR to make accurate predictions by moving the predictions in the direction of what is about to happen. This requires that the PSR learning is delayed by the length of the maximum test. A look ahead buffer is kept, allowing matching of tests to actual futures.

The final part of Algorithm 1 is to discover new core tests. Lines 18–25 propose a new core test every N iterations. A test is potentially in $\mathcal{Q}$ if its predictions over all histories $\Pr(t|H)$ are orthogonal to all other core tests $\Pr(\mathcal{Q}|H)$, i.e., if it is impossible to compute the test prediction from a linear combination of the current core tests. The orthogonality of candidates $t \in \mathcal{X}$ is measured by computing the condition number (using singular value decomposition) of a matrix made up of $\Pr(\mathcal{Q}|H)$ plus the column of predictions for the candidate test $\Pr(t|H)$. The test that results in the lowest condition number is added to the core set as the most independent test. Here we have diverged slightly from McCracken and Bowling [2006] by allowing only a single core test to be added at each time, not requiring a minimum threshold to be satisfied. We also require that N is much greater than the number of stored rows of $\Pr(\mathcal{Q}|H)$ to remove initially poor predictions of new tests. These measures avoid adding two core tests in one round that might individually be independent of the core test set, but are not independent of each other. Note that for the purposes of learning a good control policy it does not matter if non-core tests are

Algorithm 1 Online PSR Learning & Discovery

$i = 1$ $\mathcal{Q} = \{\varepsilon\}$, $\mathcal{T}$ =`expand`$(\mathcal{Q})$, α_{psr} step size
Let $\Pr(\mathcal{T}|H)$ be an $N \times |\mathcal{T}|$ matrix with rows indexed by $h_i = i \mod N$ and columns indexed by tests $t \in \mathcal{T}$.
Let $\Pr(\mathcal{Q}|H)$ be an $N \times |\mathcal{Q}|$ matrix with columns indexed by tests $q \in \mathcal{Q} \subset \mathcal{T}$.
while $|\mathcal{Q}| <$ max core tests **do**
  For action a_i, get subsequent observation o_i
  for each $\{t : a_i o_i t \in \mathcal{T}\}$ **do**
    $\Pr(t|h_i) = \frac{\Pr(a_i o_i t|h_{i-1})}{\Pr(a_i o_i|h_{i-1})}$
  end for
  for each $\{t : a_i o_i t \notin \mathcal{T}\}$ **do**
    $\mathbf{m}_t = \arg\min_{\mathbf{m}} \| \Pr(\mathcal{Q}|H)\mathbf{m} - \Pr(t|H)\|^2$
    $\Pr(t|h_i) = \Pr(\mathcal{Q}|h_i)\mathbf{m}_t$
  end for
  `normalise`$(\Pr(\mathcal{T}|h_i))$
  for each $\{tao : tao$ observed in future$\}$ **do**
    $\Pr(tao|h_i) \leftarrow (1-\alpha_{psr}) \Pr(tao|h_i) + \alpha_{psr} \Pr(t|h_i)$
  end for
  `normalise`$(\Pr(\mathcal{T}|h_i))$
  if $i \mod N = 0$ (i.e., matrix full) **then**
    for each $\{aot : t \in \mathcal{X}\}$ **do**
      $\Pr(\widehat{\mathcal{Q}}|H) = [\Pr(\mathcal{Q}|H) \ \Pr(aot|H)]$
      $\boldsymbol{\kappa}_{aot}$ = `conditionNumber`$(\Pr(\widehat{\mathcal{Q}}|H))$
    end for
    $\mathcal{Q} \leftarrow \{\mathcal{Q}, \arg\min_{aot} \boldsymbol{\kappa}_{aot}\}$
    $\mathcal{T}$ = `expand`$(\mathcal{Q})$
  end if
  $i \leftarrow i + 1$
end while

accidentally added to $\mathcal{Q}$, provided at least all the true core tests are in the final set.

Algorithm 1 is slow, solving many $O(|\mathcal{Q}|^3)$ least squares problems for each received action/observation pair. However, once the algorithm has found the maximum number of core tests we can switch to tracking the PSR prediction vector using the $O(|\mathcal{Q}|)$ update (2). This requires a final set of least squares solutions to compute parameter vectors $\mathbf{m}_t$, $\forall t \in \mathcal{X}$.

4.2 Natural Actor-Critic

The Natural Actor-Critic (NAC) algorithm [Peters *et al.*, 2005] combines policy-gradient (PG) actors, with value based critic estimates; taking advantage of policy-gradient properties of local convergence under function approximation and partial observability, with the lower variance of value based methods. Furthermore, NAC actor gradient estimates are *natural* gradients and make full use of each sample through a least-squares approach to the data. NAC is well suited to improving a policy where the observation of state is a vector basis feature. In the case of PSRs the basis feature is exactly the PSR prediction vector $\Pr(\mathcal{Q}|h_i)$. In the case of POMDP belief states the basis feature is $\mathbf{b}_i$. For convenience we use the notation $\mathbf{b}$ for both forms, meaning a *basis* feature.

The algorithm of Peters *et al.* [2005] produces noisy batch gradient estimates. We modified NAC to adjust the parameters at every step, i.e., stochastic gradient ascent. We hope to outperform batch methods because the policy can improve at every step. We also avoid the problems introduced when noisy batch gradient estimates are treated as exact and used in quasi-newton methods, line searches, or conjugate gradient methods. RL methods that use soft-max functions must be particularly careful not to step too far in any apparently good direction because the soft-max exhibits a plateau (an apparent local maximum) in the gradient for most large absolute parameters values. The natural gradient compensates for this by ensuring that gradients tell us how to step in probability space, independently of the parameterisation. We briefly describe NAC but defer to Peters *et al.* [2005] for a full explanation.

We begin with the Bellman equation for fixed parameters $\boldsymbol{\theta}$ where the value of action a in state s is $\mathbb{Q}(s,a)$. This can also be written as the value $\mathbb{V}(s)$ plus the advantage of action a, $\mathbb{A}(s,a)$:

$$\mathbb{Q}(s,a) = \mathbb{V}(s) + \mathbb{A}(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \Pr(s'|s,a)\mathbb{V}(s'). \tag{3}$$

Now $\mathbf{b}$ is the basis feature, or sufficient statistic, that reveals information about the current state s. We substitute linear approximators for the value and advantage functions, with parameter vectors $\mathbf{v}$ and $\mathbf{w}$ respectively: $\widehat{\mathbb{V}}(s) := \mathbf{b}(s)^\intercal \mathbf{v}$, $\widehat{\mathbb{A}}(s,a) := (\nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b}(s), \boldsymbol{\theta}))^\intercal \mathbf{w}$, leading to

$$\mathbf{b}(s)^\intercal \mathbf{v} + (\nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b}(s), \boldsymbol{\theta}))^\intercal \mathbf{w} = r(s,a) + \gamma \sum_{s \in \mathcal{S}} \Pr(s'|s,a)\mathbf{b}(s')^\intercal \mathbf{v}. \tag{4}$$

The surprising choice of $\nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b}(s), \boldsymbol{\theta})$ as features for estimating $\widehat{\mathbb{A}}(s,a)$ has the nice property that the parameters $\mathbf{w}$ turn out to be exactly the naturalised gradient of the long-term average reward with respect to the policy (actor) parameters [Peters *et al.*, 2005]. This is a consequence of the Policy-Gradient theorem of [Sutton *et al.*, 2000]: let $\Pr(s|\boldsymbol{\theta})$ be the steady state probability of state s and $B(s)$ is a baseline to reduce the variance of gradient estimates

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} \Pr(s|\boldsymbol{\theta}) \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \Pr(a|s)(\mathbb{Q}(s,a) - b(s)) \tag{5}$$

The obvious baseline for making $\mathbb{Q}(s,a)$ zero mean is $b(s) = \mathbb{V}(s)$, which gives $\mathbb{Q}(s,a) - \mathbb{V}(s) = \mathbb{A}(s,a)$.

Again, we substitute the linear approximation $\hat{\mathbb{A}}(s,a)$ for $\mathbb{A}(s,a)$ and make use of the fact that our policy is actually a function of $\mathbf{b} := \mathbf{b}(s)$ and $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \int_{\mathcal{S}} \Pr(s|\boldsymbol{\theta}) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \Pr(a|\mathbf{b},\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b},\boldsymbol{\theta}))^{\intercal} \mathbf{w} \, da \, ds.$$

But $\nabla_{\boldsymbol{\theta}} \Pr(a|\mathbf{b},\boldsymbol{\theta}) = \Pr(a|\mathbf{b},\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b},\boldsymbol{\theta})$ gives

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \int_{\mathcal{S}} \Pr(s|\boldsymbol{\theta}) \int_{\mathcal{A}} \Pr(a|\mathbf{b},\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b},\boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} \log \Pr(a|\mathbf{b},\boldsymbol{\theta}))^{\intercal} da \, ds \mathbf{w} =: F_{\boldsymbol{\theta}} \mathbf{w}$$

A key observation is that the matrix $F_{\boldsymbol{\theta}}$ is the outer product of the log action gradient, integrated over all states and actions. This is the Fisher information matrix. On the other hand, the naturalisation of gradients consists of pre-multiplying the normal gradient by the inverse of the Fisher matrix, leading to cancellation of the two Fisher matrices $F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = \mathbf{w}$.

NAC proceeds by solving a TD estimate of (4) for $\mathbf{w}$ and stepping the policy parameters $\boldsymbol{\theta}$ in that direction. The TD estimate replaces the summation by a sampled approximation $\gamma \mathbf{b}_i \mathbf{v}$ of the discounted value of the observed next state. This approximation introduces a zero-mean error σ. Rewriting (4) as a sampled linear system and manipulating, and defining the eligibility trace as $z_i := \sum_{j=1}^{i} \lambda^{i-j} [\nabla_{\boldsymbol{\theta}} \log \Pr(a_i|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)^{\intercal}, \mathbf{b}_i^{\intercal}]^{\intercal}$, yields

$$z_i [(\nabla_{\boldsymbol{\theta}} \log \Pr(a_i|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)^{\intercal}, (\mathbf{b}_{i-1} - \gamma \mathbf{b}_i)^{\intercal}] [\mathbf{w}_i^{\intercal}, \mathbf{v}_i^{\intercal}]^{\intercal} + \sigma = z_i r(s_i, a_i) =: \mathbf{g}_i$$

The NAC algorithm solves for $\mathbf{w}$ by averaging both sides over M steps (removing σ) and solving the resulting least squares problem $A_M [\mathbf{w}^{\intercal}, \mathbf{v}^{\intercal}]^{\intercal} = \boldsymbol{\omega}_M$, where

$$A_M = \frac{1}{M} \sum_{i=1}^{M} z_i [(\nabla_{\boldsymbol{\theta}} \log \Pr(a_i|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)^{\intercal}, (\mathbf{b}_{i-1} - \gamma \mathbf{b}_i)^{\intercal}], \tag{6}$$

and $\boldsymbol{\omega}_i = \frac{1}{M} \sum_{i=0}^{M} \mathbf{g}_i$.

Algorithm 2 is our online version of NAC. The main difference from the original is the avoidance of a $O(d^3)$ matrix inversion for solving $A_M [\mathbf{w}^{\intercal}, \mathbf{v}^{\intercal}]^{\intercal} = \boldsymbol{\omega}_M$, where $d = |\boldsymbol{\theta}| + |\mathbf{b}|$. Instead, we implement the Sherman-Morrison rank-1 matrix inverse update, with complexity $O(d^2)$. As the number of parameters and the basis vector grow, NAC still becomes prohibitive. For example, a tabular representation of policy, such as used for k-step finite-memory experiments, needs $(|\mathcal{O}||\mathcal{A}|)^k |\mathcal{A}|$ parameters to represent the exact policy and length $(|\mathcal{O}||\mathcal{A}|)^k$ basis vectors. Even our online version of NAC would then require $O(d^2) = O((|\mathcal{O}||\mathcal{A}|)^{2k})$ steps *per iteration*. So our finite memory and other tabular representation experiments used a simplified $O(d)$ version of the algorithm that sets A to the identity matrix, equivalent to the GPOMDP algorithm of Baxter *et al.* [2001]. NAC can be viewed as a generalisation of GPOMDP that incorporates curvature information to accelerate convergence, without changing the optimisation problem.

Algorithm 2 An Online Natural Actor-Critic

1: $i = 1$, $A_1^{-1} = I$, $\boldsymbol{\theta}_1 = [0]$, $\mathbf{z}_1 = [0]$
2: α_{pg}=step size, γ=Critic discount, λ=Actor discount
3: Get observation $\mathbf{b}_0$
4: **while** not converged **do**
5: Sample action $a_i \sim \Pr(\cdot|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)$
6: $\mathbf{z}_i = \lambda \mathbf{z}_{t-1} + [\nabla_{\boldsymbol{\theta}} \log \Pr(a_i|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)^{\intercal}, \mathbf{b}_i^{\intercal}]^{\intercal}$
7: Do action a_i
8: Get reward r_i
9: $\mathbf{g}_i = r_i \mathbf{z}_i$
10: Get updated statistic $\mathbf{b}_i$
11: $\mathbf{y}_i = [\nabla_{\boldsymbol{\theta}} \log \Pr(a_i|\mathbf{b}_{i-1},\boldsymbol{\theta}_i)^{\intercal}, \mathbf{b}_{i-1}^{\intercal}]^{\intercal} - \gamma [0^{\intercal}, \mathbf{b}_i^{\intercal}]^{\intercal}$
12: $\mu_i = 1 - \frac{1}{t}$
13: $\mathbf{u}_i = (1 - \mu_i) A_{t-1}^{-1} \mathbf{z}_i$
14: $\mathbf{q}_i^{\intercal} = \mathbf{y}_i^{\intercal} A_{t-1}^{-1}$
15: $A_i^{-1} = \frac{1}{\mu_i} A_{t-1}^{-1} - \frac{\mathbf{u}_i \mathbf{q}_i^{\intercal}}{1.0 + \mathbf{q}_i^{\intercal} \mathbf{z}_i}$
16: $[\mathbf{w}_i^{\intercal}, \mathbf{v}_i^{\intercal}]^{\intercal} = A_i^{-1} \mathbf{g}_i$
17: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_i + \alpha_{pg} \mathbf{w}_i$
18: $i \leftarrow i + 1$
19: **end while**

4.3 Policy Architecture

The policy maps history statistics $\mathbf{b}_i$ to *distributions* $\tilde{\mathbf{a}}_i$ over actions using a function f with outputs $\mathbf{x}$ and the soft-max function. We implemented two versions of f: the first is an exact tabular representation where $\mathbf{x}_i = f(a_i o_i \ldots a_{i-k-1} o_{i-k-1})$ is a row of parameter matrix $\boldsymbol{\theta}$, indexed by the last k steps of history. The second maps vectors (Figure 1), either POMDP beliefs or PSR prediction vectors, to $\mathbf{x}$ using a linear approximator $\mathbf{x} = \boldsymbol{\theta}\mathbf{b}$. In this case $\boldsymbol{\theta}$ is an $|\mathcal{A}| \times |\mathcal{S}|$ parameter matrix. Now let $\mathbf{U}(a)$ be the unit vector with a 1 in row a. Thus, assuming element-wise exponentiation

$$\tilde{\mathbf{a}}_i = \frac{\exp(\mathbf{x}_i)}{\sum_{a \in \mathcal{A}} \exp(\mathbf{x}_i(a))}; \quad \Pr(a_i = a'|\mathbf{b}_{i-1},\boldsymbol{\theta}_i) = \tilde{\mathbf{a}}_i(a');$$
$$\nabla_{\boldsymbol{\theta}_i} \log \Pr(a_i = a'|\mathbf{b}_{i-1},\boldsymbol{\theta}_i) = (\mathbf{U}(a') - \tilde{\mathbf{a}}_i) \mathbf{b}_i^{\intercal}. \tag{7}$$

5 Experiments

We experimented on 6 small POMDP benchmarks commonly used in PSR papers [James *et al.*, 2004; Bowling *et al.*, 2006]. They range from 2 to 11 states,

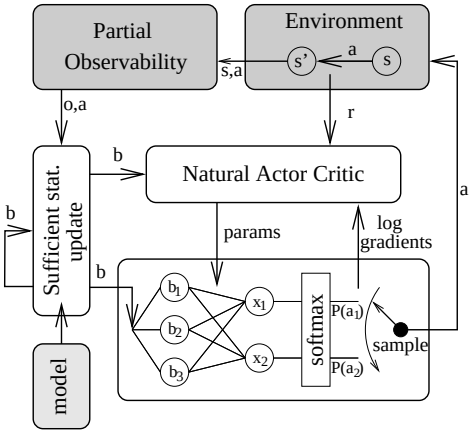

Figure 1: A linear approximator architecture for mapping sufficient statistic **b** to stochastic policies.

and 2 to 7 observations. All experiments used the discount factors defined in the scenario files. We hand-tuned α_{pg} to the best power of 10 for each scenario. Our benchmark algorithms are split into two categories: those with policies that depend explicitly on zero or more steps of history and use an exact tabular policy parameterisation; and those that use sufficient statistic vectors and a linear policy parameterisation. The exception was PSR experiments on Shuttle. This problem required a multi-layer perceptron with 3 hidden units using a tanh non-linearity and initial parameters randomly initialised to $\|\boldsymbol{\theta}\|_\infty < 0.01$. We comment why this was necessary later. The tabular algorithms include:

Random: the policy is a uniform distribution over all actions. No learning occurs.

Blind: completely unobservable. The policy is a stationary distribution over actions.

Reactive: policies depend only on o_i.

Finite Memory: policies depend on k past (o_i, a_i) pairs, with $k = 2$.

MDP: policies depend on the true state of the system. This algorithm provides an upper-bound on the obtainable long-term reward.

The NAC algorithm is not known to work with basis vectors that are a non-deterministic indication of state. This reason, and the high complexity of NAC with exact policy representation, led us to use the vanilla gradient algorithm for these runs. The exact policy representation gives these algorithms an advantage, but would not be scalable to larger problems due to the exponential explosion in parameters with growing history length. The results are given in Table 1. Standard deviations over 30 trials are also given. On a 3GHz Pentium IV desktop the average time required to reach 95% of maximum R was from 2 seconds per run (reactive Tiger) up to 180 seconds (finite-memory 4x3 Maze). Similarly, the number of learning steps varied from $1.3 \cdot 10^6$ (reactive Tiger) to $1.6 \cdot 10^7$ (finite-memory 4x3 Maze).

The more interesting experiments involve learning a policy from sufficient statistics, exact or estimated. We expected that they would out-perform the reactive approach, while *not* expecting them to beat the fully observable MDP case.

Estimated POMDP: the underlying POMDP parameters are estimated using a modified Baum-Welsh style EM algorithm [Chrisman, 1992]. Re-estimation is performed every N steps using a history of N oa-pairs (with $N = 1000$). The model is initialised with the correct number of states, and random values in the transition and observation tables. There are independent transition and observation matrices for each action, allowing the actions to drive the evolution of the estimated belief. The belief state is updated with (1).

Estimated PSR: discovery was performed up to the maximum number of core tests shown in Table 2. For the smaller problems we overestimate the number of core tests, allowing for some error in discovery. For larger scenarios the discovery struggled to find more than the indicated number of core tests. Learning was performed with a step size of $\alpha_{psr} = 1$, giving the full probability mass to the successful test. Smaller values of α_{psr} performed poorly in practice. This occurs because for $\alpha_{psr} < 1$ the momentum term does not allow test probabilities to fully commit to their parent test values. Such a high α does not necessarily degrade the final result because the final PSR parameters come from a least squares estimation that finds a compromise solution that best matches all the probabilities in the history matrix.

Exact POMDP: the true POMDP model is used to compute the belief state using (1).

Exact PSR: the exact PSR model is computed from the true POMDP model using the depth first search algorithm presented in Littman *et al.* [2002]. We confirmed the correctness of the core tests by comparing with previous papers. The only discrepancy was the Paint problem where we found only 2 core tests, agreeing with McCracken and Bowling [2006], but disagreeing with the 4 reported by James *et al.* [2004].

Table 2 gives the results.[1] The immediate conclusion is that all statistics are successful in improving upon

[1]The results were generated using the LibPG policy-gradient toolkit: `http://sml.nicta.com.au/~daa/software.html`.

the reactive policy except for the estimated PSR for Tiger and Paint, and the exact PSR model for Paint. The Paint scenario seems to be ill-conditioned from a PSR point of view, possibly because the observation is almost always "not blemished", so any two tests will appear almost linear. This was confirmed by observing that the SVD of the Paint history matrix produced only 2 significant singular values, and more values on the edge of machine precision. The other difficult scenario, Tiger, suffers from a poor local maximum which is to always "listen", avoiding the large penalty for opening the wrong door. This, combined with noisy estimation of the PSR parameters, means policies get stuck in this poor maximum.

Empirically we observed that it was generally more difficult to *learn* a policy from PSR prediction vectors than from belief states. This is despite the fact that PSRs have provably at *least* as much representational power as POMDPs. This led to a simple result about using belief states in conjunction with perceptrons

Proposition 1. *Let* $\Pr(a|\mathbf{b}_i, \boldsymbol{\theta})$ *be parameterised as a linear transformation* $\mathbf{x}_i = \boldsymbol{\theta}\mathbf{b}_i$*, as input to a soft-max function (7). Also let* $a^* = \pi(s)$ *be the optimal deterministic policy given full observability. Then* $\boldsymbol{\theta}$ *can always be constructed such that as* $\|\boldsymbol{\theta}\| \to \infty$*, the policy performance approaches that of a POMDP "voting" heuristic* $a_i = \arg\max_a \sum_{\{s:\pi(s)=a\}} \mathbf{b}(s)$.

Proof. We prove this by constructing the parameters $\boldsymbol{\theta}$ that implement the voting policy. We set all parameters in row $\boldsymbol{\theta}_a$ to 0 except for the entries $\boldsymbol{\theta}_{\pi(s),s}$ which we set to $\chi \to \infty$. Now the output of the linear approximator for each action is $\mathbf{x}(a) = \sum_{\{s:\pi(s)=a\}} \chi \mathbf{b}(s)$. Because the soft-max function uses the ratio of *exponentials* of $\mathbf{x}(a)$, a sufficiently large χ turns the soft-max into a hard-max, always choosing the action voted for by the states with the largest sum of beliefs. □

Thus, a linear transform of a belief state to soft-max input can always represent a policy *at least* as good, and possibly better, than the POMDP voting heuristic policy [Simmons and Koenig, 1995]. Of course, the existence of a set of parameters that represents a good policy does not imply that it will be found by NAC, it does however suggest a lower bound on the quality of linearly parameterised policies we should achieve with good sufficient statistics. Unless it is known that each core tests suggests only states with a common optimum action, we cannot easily construct a similar linear parameterisation for PSR prediction vectors.

Put another way, knowing the true MDP state always allows you to compute a direct mapping to an optimal action. Since the POMDP belief state is a direct approximation of which state you are in, you can always compute a soft-max mapping that chooses the best action voted for by the most likely states in the belief vector. This also explains our experience with PSRs and the Shuttle problem. We could not do better than a reactive policy without introducing hidden layers into the parameterisation, allowing a much richer mapping from the sufficient statistic vector to a policy.

Learning is particularly difficult with *estimated* PSR parameters, with averaged results significantly worse than the exact PSR experiments (but still better than reactive). Figure 2 shows the difference in convergence between the estimated PSR and exact PSR on the Network problem. Improving Algorithm 1 by reducing the errors from least squares estimates and gradient steps would help significantly. The fastest of these algorithms was the exact belief state, requiring on average from 2 seconds (Tiger) to 350 seconds (Cheese). The slowest was the estimated PSR algorithm, requiring between 120 seconds (Tiger) to 8000 seconds (Cheese). The majority of the time was taken up in the online PSR discovery and learning. In terms of the number of iterations required NAC is an order of magnitude faster than the tabular vanilla PG algorithm used for Table 1. The average number of iterations ranged from best case $8.6 \cdot 10^4$ (exact POMDP Tiger) to $3.8 \cdot 10^6$ (exact PSR Maze). Thus, if experience is more expensive than computation time then NAC is attractive, otherwise vanilla estimates are faster.

Table 1 shows that these common PSR benchmark domains can actually be solved quite well using finite history methods. This is largely because this amount of history suffices for most of these problems. Also, tabular methods have the advantage of an *exact* policy representation. The Tiger problem is interesting because the finite history methods fails compared to PSRs, exactly because it is a problem that requires more than two observations to do well.

Our experiments perform model estimation and policy improvement separately. We also attempted to do this simultaneously.[2] This can be dangerous if the policy finds a poor local maximum before giving the estimation algorithm sufficient time to explore the full system dynamics. However, for the Network and Shuttle problems we achieved average values of 12 and 1.3 respectively. Simultaneous learning on the other scenarios led only to the reactive policy.

[2] A modification to the PSR estimation is necessary to allow the policy to get a PSR prediction vector based on the most recent observation, while still allowing the PSR to use a look-ahead buffer for making TD steps. Our modification was to wind forward the PSR estimate through all actions and observations in the look-ahead buffer using the most recently computed parameter vectors $\mathbf{m}_t \; \forall t \in \mathcal{T}$.

Table 1: Baseline results with tabular representations.

Scenario	Tiger	Paint	Network	Shuttle	4x3Maze	Cheese
Rnd	-30	-.20	-12	-0.2	-.10	.01
Blind	-1.0± .00	.00± .000	-1± 0.2	0.4± .01	.05± .001	.01± .001
React	-1.0± .00	.00± .000	10± 0.2	1.1± .01	.09± .001	.11± .001
Hist	0.8± .49	.18± .003	14± 1.0	1.8± .02	.14± .002	.19± .002
MDP	10± .00	.65± .001	26± 0.3	1.8± .00	.18± .001	.20± .001

Table 2: Final results over 30 runs for sufficient statistics algorithms. Numbers in brackets show the true and discovered number of core tests for each scenario.

Scenario	Tiger	(2)	Paint	(4)	Network	(7)	Shuttle	(8)	4x3Maze	(11)	Cheese	(11)
Est.Belief	0.4± 1.0		.11± .021		14± 0.4		1.8± .05		.15± .003		.16± .011	
Est.PSR	-1.5± 1.5	(3)	.01± .012	(3)	12± 1.5	(8)	1.6± .23	(8)	.11± .007	(10)	.14± .030	(8)
Belief	1.4± .033		.13± .001		13± 0.7		1.8± .03		.14± .001		.16± .006	
PSR	1.2± .529	(2)	.00± .000	(2)	14± 0.6	(7)	1.7± .16	(8)	.12± .019	(11)	.16± .002	(11)

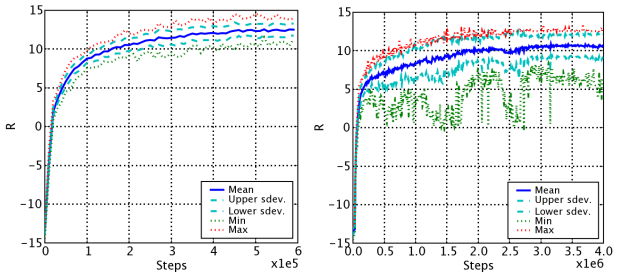

Figure 2: Exact PSR predictions (top) are easier to learn from than Estimated PSR predictions (bottom).

6 Conclusion

We have demonstrated that tracking sufficient statistics with exact and estimated models can be used to learn significantly better than reactive policies. Finite memory statistics also work well, but will not scale well to problems with more actions, observations, and greater history. NAC was effective at learning from vector representations of state. Future work will improve online PSR estimation, exploring the usefulness of non-linear PSR representations.

Acknowledgments

National ICT Australia is funded by the Australian Government's Backing Australia's Ability program and the Centre of Excellence program. This project was also funded by the Australian Defence Science and Technology Organisation.

References

J. Baxter, P. Bartlett, and L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *JAIR*, 15:351–381, 2001.

M. Bowling, P. McCracken, M. James, J. Neufeld, and D. Wilkinson. Learning predictive state representations using non-blind policies. In *Proc. ICML 2006*. AAAI Press, 2006.

L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *National Conference on Artificial Intelligence*, 1992.

M. R. James, S. Singh, and M. Littman. Planning with predictive state representations. In *Proc. ICMLA-04*, pages 304–311, 2004.

M. Littman, R. Sutton, and S.Singh. Predictive representations of state. In *Proc. NIPS'01*, volume 14. MIT Press, 2002.

A. McCallum. *Reinformcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

P. McCracken and M. Bowling. Online discovery and learning of predictive state representations. In *Proc. NIPS'05*, volume 18. MIT Press, 2006.

J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Proc. ECML*. Springer-Verlag, 2005.

E. J. Rafols, M. Ring, R. Sutton, and B. Tanner. Using predictive representations to improve generalization in reinforcement learning. In *Proc. IJCAI*, pages 835–840, 2005.

R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. IJCAI*, 1995.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proc. UAI 2004*, pages 512–519, 2004.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. NIPS'99*, volume 12. MIT Press, 2000.