INSTITUT NATIONAL DE LA STATISTIQUE ET DES ETUDES ECONOMIQUES
Série des Documents de Travail du CREST
(Centre de Recherche en Economie et Statistique)

n° 2010-36

Bayesian Learning of Noisy Markov Decision Processes

S. S. SINGH[1] - N. CHOPIN[2]
N. WHITELEY[3]

[1] University of Cambridge, Department of Engineering and Statistical Laboratory, United Kingdom. Email : sss40@cam.ac.uk

[2] CREST – ENSAE, 3 Avenue Pierre Larousse, 92240 Malakoff, France. Email : nicolas.chopin@ensae.fr

[3] University of Bristol, Department of Statistics, United Kingdom : Email : nick.whiteley@bristol.ac.uk

Bayesian learning of noisy Markov decision processes

Sumeetpal S. Singh[*], Nicolas Chopin[†], Nick Whiteley[‡]

27 July 2010

Abstract

This work addresses the problem of estimating the optimal value function in a Markov Decision Process from observed state-action pairs. We adopt a Bayesian approach to inference, which allows both the model to be estimated and predictions about actions to be made in a unified framework, providing a principled approach to mimicry of a controller on the basis of observed data. A new Markov chain Monte Carlo (MCMC) sampler is devised for simulation from the posterior distribution over the optimal value function. This step includes a parameter expansion step, which is shown to be essential for good convergence properties of the MCMC sampler. As an illustration, the method is applied to learning a human controller.

Key-words: Markov Chain Monte Carlo, Data augmentation, Parameter expansion.

1 Introduction

1.1 Motivation

The problem of fitting a statistical model to observed actions has received significant attention in a variety of disciplines. These include Optimal Control (Rust 1988), Economics (Gotz and McCall 1980; Wolpin 1984; Rust 1987; Hotz and Miller 1993; Geweke and Keane 2000; Geweke et al. 1994; Aguirregabiria and Mira 2002; Imai et al. 2009), and Machine Learning (Ng and Russell 2000; Abbeel and Ng 2004). Across these cases there is some variety in the estimation

[*]Department of Engineering and Statistical Laboratory, University of Cambridge, United Kingdom. E-mail: sss40@cam.ac.uk

[†]CREST—ENSAE, F-92245 Malakoff cedex, France. Email: Nicolas.Chopin@ensae.fr

[‡]Department of Statistics, University of Bristol, United Kingdom. Email: Nick.Whiteley@bristol.ac.uk

aims and the assumed mechanisms which generate observed actions. We focus on the case in which it is assumed that the observations arise from an underlying Markov Decision Process (a full model specification is given in the next section). In this case, given the current state X of the system, the controller chooses an action A and receives an instantaneous reward specified by the function r,

$$(X, A) \to r(X, A) \in \mathbb{R}$$

where (X, A) is the state-action pair. The state then evolves according to a Markov kernel $p(\cdot|X, A)$, the controller chooses the next action, receives a reward and so on. The controller chooses its actions to maximize the average reward it will accrue over an infinite horizon. However, the controller may take sub-optimal decisions from time to time. This is captured by a "noisy" MDP model.

A generic approach to automating a task is to model it as a control problem (Bertsekas and Tsitsiklis 1996, Chapter 8), which involves specifying a reward function and other elements of the model, and then solving for the optimal controller. Putting aside the difficulties associated with the last step, specifying the reward function is non-trivial and often achieved in practice by a heuristic process of trial and error, i.e. by observing the system under the computed optimal controller and then adjusting the reward function to avoid observed undesired behavior. After this adjustment, the optimal controller is re-computed and this process is repeated until satisfaction.

It is possible to simplify this automation process. In particular, it is often possible to obtain a sample path which is characteristic of desired behavior, e.g. by a human controller, and then estimate the control policy generating this sample path: this is known as the inverse reinforcement learning problem (Ng and Russell 2000; Abbeel and Ng 2004). Learning to mimic a controller has many potential applications in applied fields such as robotics and artificial intelligence (Coates et al. 2009); biology, e.g. the study of animal learning (Watkins 1987; Schmajuk and Zanutto 1997), economics (Rust 1987), and other fields.

Our aim is to develop a purely statistical, and computationally tractable, solution to this problem, based on a statistical model for the controller's actions, and the Bayesian approach.

This is advantageous because it gives us a unified and principled framework for the following tasks: (a) to properly model uncertainty (i.e. the human controller may make mistakes); (b) to estimate jointly the control policy and the model parameters, (c) to predict future actions.

In a parametric approach to estimation, the reward function and other elements of the model are assumed to be specific functional forms of a parameter vector θ (Gotz and McCall 1980; Wolpin 1984; Rust 1987, 1988; Aguirregabiria and Mira 2002; Hotz and Miller 1993; Imai et al. 2009). The best parametric estimate may then be computed, for example, by maximizing the likelihood of the observed data with respect to θ. From a computational perspective, we shall see that this approach is cumbersome for a noisy MDP model because the likelihood of an observed action given observed states is a complicated integral. We avoid this difficulty by targeting the control policy directly; as we shall see, a specific quantity called the optimal value function. This gives an additional justification to the Bayesian approach in this context, as the data augmentation principle (Tanner and Wong 1987) makes it possible to estimate the model without computing the difficult integral mentioned.

1.2 Contributions

The contributions of this work are as follows. We adopt a Bayesian approach and infer the optimal value function of noisy MDPs. Our approach is inspired by the pioneering work of Albert and Chib (1993), McCulloch and Rossi (1994) in the context of statistical inference in discrete choice models, where the computations are performed using a Gibbs sampler applied to an enlarged (or augmented) model. In subsequent work, Nobile (1998), Imai and van Dyk (2005) enhanced the computational efficiency of the Gibbs sampling technique while McCulloch et al. (2000), Imai and van Dyk (2005), devised new priors for the identified parameters of the model.

We devise a new Gibbs sampling algorithm for inferring the optimal value function. The proposed algorithm is a Parameter Expanded Data Augmentation (PX-DA) algorithm (Liu et al. 1999; Meng and van Dyk 1999). PX-DA improves upon the efficiency of standard DA by reducing the correlation between the samples. This is achieved by inserting an additional

simulation step into the algorithm which involves moving in the augmented data space. This extra simulation step is computationally inexpensive and leads to improved performance over standard DA algorithms. In fact, we give examples where the DA algorithm does not converge whereas PX-DA does. The algorithms of Nobile (1998), Imai and van Dyk (2005) are indeed PX-DA type samplers and ours differs in several ways. Particularly, the augmented data is moved in the extra simulation step with a different transformation and with an improper prior for the marginal distribution over the parameters of the transformations. This is done to enhance mixing. We also implement an efficient Metropolis-Hastings kernel with independent proposals when sampling the augmented data.

As an illustrative example of learning a human controller we apply our framework to the game of Tetris. Automating Tetris is a challenging benchmark problem in the control literature, see for example Bertsekas and Tsitsiklis (1996), and treating it is difficult because the control model has a very large state space. Moreover, data from a human player is noisy, as we are prone to making errors. We show the proposed method can quite accurately mimic the human player by performing posterior prediction from a limited amount of observed data. By contrast, existing approaches from the control literature focus solely on, having specified the reward function, solving the associated difficult optimization problem using reinforcement learning techniques (Tsitsiklis and Roy 1994; Bertsekas and Tsitsiklis 1996).

1.3 Plan, notation

The organization of this paper is as follows. Section 2 defines the problem in detail and states the inference objectives. Section 3 describes the PX-DA method generally and then the specific implementation for inferring the value function. Section 4 presents the PX-DA sampler in detail for the assumed priors and discusses some practical issues and extensions. Numerical results highlighting various properties of the proposed PX-DA algorithm are presented in Section 5, as well as implementation details and results for Tetris. A proof of the correctness of the proposed PX-DA algorithm is presented in the Appendix along with implementation details of the MCMC algorithm.

This section is concluded with a description of the notation used. Capital letters are used for random variables and lower case for their realizations. The letters f and p are reserved for the probability densities or probability mass functions of random variables. For two jointly distributed random variables (X, Y), $f_{X|Y}$, $f_{X,Y}$ and f_X denote, respectively, the conditional probability density, the joint density and the marginal density. When the subscript is omitted, the arguments of f or p will indicate precisely the random variables to which the density corresponds. For example, $p(x|y)$ is $p_{X|Y}(x|y)$. $\mathcal{N}(x; \mu, \Sigma)$ is the value at x of the multivariate normal probability density with mean μ and covariance Σ. For a vector v, the i-th component is denoted $v(i)$. All vectors are column vectors and the transpose of v is indicated by v^{T}. The m-dimensional vector comprised of ones (respectively zeros) only is denoted by $\mathbf{1}_m$ (respectively $\mathbf{0}_m$). The subscript m is omitted when the dimension is obvious from context. I_m will denote the m by m identity matrix. Similarly, $[\Sigma]_{i,j}$ will denote the (i, j)-th element of the matrix Σ. $\mathbb{I}_A$ is the indicator function of the set A, i.e. $\mathbb{I}_A(x) = 1$ if $x \in A$ and 0 otherwise. $\mathbb{R}$ denotes the real line, $\mathbb{R}_+$ its strictly positive part and $\mathbb{E}$ is the mathematical expectation operator. The cardinality of a finite set A is denoted by $|A|$. The Dirac measure concentrated at a point x is denoted by δ_x.

2 Problem Statement

2.1 Markov decision processes

A MDP is comprised of a controlled Markov chain, a control process, a reward function and an optimality criterion. Each of these are defined in turn below; see Bertsekas and Tsitsiklis (1996) for additional background.

The state process, denoted $\{X_k\}_{k\geq 0}$, is a $\mathcal{X}$-valued controlled Markov chain where $\mathcal{X}$ is the finite set $\{1, 2, \ldots, N\}$. Let $\{A_k\}_{k\geq 0}$ be the $\mathcal{A}$-valued control (or action) process where $\mathcal{A} = \{1, 2, \ldots, M\}$ is the set of all possible controls. Given the entire realization of the state and actions up to time k, the evolution of the state to time $k+1$ is determined by the selected

action and state at time k only, i.e.

$$X_{k+1}|\left(X_{0:k}=x_{0:k},\, A_{0:k}=a_{0:k}\right)\sim p(\cdot|x_k,a_k), \tag{1}$$

where for each state-action pair (x,a), $p(\cdot|x,a)$ is a probability distribution on $\mathcal{X}$. The evolution of the action process is determined by a policy μ which is a mapping from the set of actions to the set of states. Particularly,

$$A_k|\left(X_{0:k}=x_{0:k},\, A_{0:k-1}=a_{0:k-1}\right)\sim \delta_{\mu(x_k)}(\cdot).$$

Let r be a real valued function on $\mathcal{X}$ which is called the reward function. The reward at time k for being is state X_k is $r(X_k)$. The optimality criterion we consider is the following discounted sum of accumulated rewards over an infinite horizon,

$$V_\mu(x_0)=\mathbb{E}_\mu\left[\sum_{k=0}^{\infty}\beta^k r(X_k)\middle| X_0=x_0\right] \tag{2}$$

where $\beta\in(0,1)$ is the discount factor ensuring the expectation is well defined.(If there exists a zero reward state which is absorbing, and all policies lead to this state with probability one for all initial states x_0 then, the expectation is well defined without the discount β.) The subscript on the expectation operator denotes the policy controlling the evolution of $\{X_k\}_{k\geq 0}$. Let μ^* be a policy satisfying the following inequality,

$$V_{\mu^*}(x_0)\geq V_\mu(x_0)\qquad \text{for all } (\mu,x_0).$$

Then μ^* is said to be optimal. It is well known that μ^* is characterized by the real valued function on $\mathcal{X}$, denoted V^*, which satisfies the following fixed point equation,

$$V^*(x)=\max_{a\in\mathcal{A}}\left\{r(x)+\beta\sum_{x'\in\mathcal{X}}p(x'|x,a)V^*(x')\right\}. \tag{3}$$

In the literature on MDPs (Bertsekas and Tsitsiklis 1996), V^* is referred to as the (optimal) value function. Given V^*, the optimal policy μ^* is

$$\mu^*(x)=\arg\max_{a\in\mathcal{A}}\left\{\sum_{x'\in\mathcal{X}}p(x'|x,a)V^*(x')\right\} \tag{4}$$

for all $x\in\mathcal{X}$.

2.2 A statistical model for imperfect policy execution

For each $a \in \mathcal{A}$, let P_a be the transition probability matrix with elements

$$[P_a]_{i,j} = p(j|i,a).$$

We consider the following statistical model for the action component a_k of each observed state-action pair (x_k, a_k) built around the MDP framework. It is assumed that

$$A_k = \arg\max_{a \in \mathcal{A}} \left\{ \epsilon_k(a) + \left(P_a V^*\right)(x_k) \right\}. \tag{5}$$

where the ϵ_k's, $k \geq 0$, are independent and identically distributed M-dimensional Gaussian variates,

$$\epsilon_k^{\mathrm{T}} = (\epsilon_k(1), \ldots, \epsilon_k(M)) \sim \mathcal{N}(\mathbf{0}_M, \Sigma).$$

The inclusion of this noise process renders the model more versatile. It may be interpreted in two different ways. First, if there are several actions that are near optimal, in the sense quantified by the numerical value of the expression in the right hand side of (4), then the controller could have selected one of the near optimal actions in error. Thus while the policy is optimal, the execution of the policy is subject to disturbance. Second, it can be shown that (5) characterizes the optimal policy of a MDP with a mixed discrete-continuous state process, $(x, \epsilon) \in \mathcal{X} \times \mathbb{R}^M$, and reward function $r : \mathcal{X} \times \mathbb{R}^M \times \mathcal{A} \to \mathbb{R}$ given by $r(x, \epsilon, a) = r(x) + \epsilon(a)$.

Given the state (x_k, ϵ_k) at time k, and action a_k, the discrete component of the next state, X_{k+1}, is drawn from (1) while the continuous component ϵ_{k+1} is drawn from $\mathcal{N}(\mathbf{0}_M, \Sigma)$. It follows from this separation in the evolution of the state components that there exists a vector $V^* \in \mathbb{R}^{|\mathcal{X}|}$ such the optimal policy for this MDP is given by (5) (Rust 1988, Theorems 3.1, 3.3). In this model, the statistician only observes the discrete component of the state process and the action taken at each time, while ϵ_k is the unobserved random component of the reward known only to the decision maker.

The data d consist of a sequence of state-action pairs, $d = \{d_k\}_{k=1}^T = \{(x_k, a_k)\}_{k=1}^T$ observed for T epochs and the aim is to infer V^*. It is assumed that the law of the controlled process, which is specified by the collection of transition matrices $\{P_a\}_{a \in \mathcal{A}}$ is known, but the reward

function r is unknown. This implies (3) cannot be used to solve for V^*. The approach below can be generalized to the case when $\{P_a\}_{a\in\mathcal{A}}$ is unknown. However, assuming $\{P_a\}_{a\in\mathcal{A}}$ is known is reasonable in a number of applications, in particular the human controller example studied in Section 5.

In Bayesian setting, a prior for V^* is chosen and inference will be based on samples from the posterior $p_{V|D}(v|d)$, henceforth denoted as $p(v|d)$ (The specification of the prior over the value function is postponed to Section 4). These samples may then also be used via (5) to estimate the optimal policy μ^* and thus predict the behavior of the system. In the context of the human controller example, d consists of the observed actions of a person.

The likelihood of the observed data is

$$p(d\,|V,\Sigma) = \prod_{i=1}^{T} p(x_i|x_{i-1},a_{i-1})p(a_i\,|V,\Sigma,x_i) \propto \prod_{i=1}^{T} p(a_i\,|V,\Sigma,x_i)$$

where, abusing notation, $p(x_0|x_{-1},a_{-1})$ denotes the prior distribution for X_0. In the second line, the terms $p(x_i|x_{i-1},a_{i-1})$ are omitted as they have no bearing on the desired posterior.

For each state $x\in\mathcal{X}$, let R_x be the time homogeneous $M\times N$ matrix

$$[R_x]_{k,l} = p(x_i = l|x_{i-1} = x, a_{i-1} = k). \tag{6}$$

The likelihood $p(a_i\,|v,\Sigma,x_i)$, or CCP, is the intractable integral

$$p(A_i = a_i\,|v,\Sigma,x_i) = \int_{\{\epsilon\in\mathbb{R}^M:\,\epsilon(a_i)\geq\epsilon(j),j\neq a_i\}} \mathcal{N}(\epsilon;R_iv,\Sigma)d\epsilon. \tag{7}$$

In the above expression, R_{x_i} has been abbreviated to R_i. Henceforth $p(A_i = a_i\,|v,\Sigma,x_i)$ will be abbreviated to $p(a_i\,|v,\Sigma)$. The likelihood is invariant to both translations of the vector v and multiplications of it by positive scalars,

$$p(d\,|v,\Sigma) = p(d\,|\sqrt{z_1}(v+z_2\mathbf{1}),z_1\Sigma),\quad \forall(z_1,z_2)\in\mathbb{R}_+\times\mathbb{R}. \tag{8}$$

The design of the PX-DA algorithm presented in the following Section is based on this property.

The assumed model for the noise corrupting the action selection process results in a target distribution similar to the multinomial probit (MNP) problem (Albert and Chib 1993; Geweke et al. 1994; McCulloch and Rossi 1994; McCulloch et al. 2000; Imai and van Dyk 2005) and

the stated invariance of the likelihood to scaling (z_1) is well documented in this literature. In the context of MNP models, the main existing approach to ensure the posterior is well defined for improper priors is to constrain enough parameters of the model to ensure identifiability of the remaining ones and then introduce priors for them. For example, by setting the last component of V to zero and then introducing a suitable prior for the remaining $N-1$ non-zero components. We shall use a slightly different approach as detailed in Section 4.

3 The PX-DA Method

Let $f_X : \mathbb{R}^p \to \mathbb{R}$ be a the target probability density from which samples are sought. In many applications, it is not possible to simulate from f_X directly. However, it is often possible to introduce a random vector $Y \in \mathbb{R}^q$ which is jointly distributed with X such that sampling from the conditional densities $f_{X|Y}$ and $f_{Y|X}$ is straightforward. This is the principle of *data augmentation* (DA) (Tanner and Wong 1987). Simulating from these densities sequentially as follows,

$$Y_{n+1}|X_n = x_n \sim f_{Y|X}(\cdot|x_n), \quad X_{n+1}|Y_{n+1} = y_{n+1} \sim f_{X|Y}(\cdot|y_{n+1}), \quad n \geq 0, \tag{9}$$

results in a Markov chain $\{X_n\}_{n\geq 0}$ with the correct asymptotic distribution for any initial state x_0 (under weak regularity assumptions) Tanner and Wong (1987):

$$\lim_{n\to\infty} \mathbb{P}(X_n \in A) = \int_A f_X(x)dx. \tag{10}$$

As noted by Liu et al. (1999), Meng and van Dyk (1999) in some situations it is possible to improve the efficiency of this sampler by introducing auxiliary variables. This technique was termed *parameter expanded* DA by Liu et al. (1999).

Let $\Lambda \subseteq \mathbb{R}^d$ and let $\{\varphi_\lambda\}_{\lambda\in\Lambda}$ be a class of one-to-one differentiable functions mapping $\mathbb{R}^q$ to itself. Let

$$J_\lambda(y') = \left| \det \left(\left[\left. \frac{\partial \varphi_{\lambda,i}(y)}{\partial y(j)} \right|_{y=y'} \right]_{i,j} \right) \right| \tag{11}$$

where $\varphi_{\lambda,i}(y)$ is i-th component function of $\varphi_\lambda(y)$. J_λ is the Jacobian determinant of the mapping $\varphi_\lambda : \mathbb{R}^q \to \mathbb{R}^q$. Let Z be a random vector in $\Lambda \subseteq \mathbb{R}^d$ with probability density f_Z. The aim is to

reduce the auto-correlation between X_n and X_{n+1} generated by the Gibbs sampler and PX-DA achieves this by inserting the an extra simulation step as follows.

Algorithm 1 ***A generic PX-DA sampler***

Given $X_n = x_n$, $Y_n = y'_n$ at iteration $n+1$, perform the following steps to sample X_{n+1}:

Step 1. Sample Y_{n+1} from $f_{Y|X}(\cdot|x_n)$ and call the sampled value y_{n+1}. (If exact sampling from $f_{Y|X}$ is not possible, sample Y_{n+1} from a Markov kernel that leaves $f_{Y|X}(\cdot|x_n)$ invariant.)

Step 2a. Sample $Z_{n+1}^{(1)}$ from $f_Z(\cdot)$, call the sample $z_{n+1}^{(1)}$ and let $\widetilde{y}_{n+1} = \varphi^{-1}_{z_{n+1}^{(1)}}(y_{n+1})$.

Step 2b. Sample another Λ-valued random variable, $Z_{n+1}^{(2)}$, from the density which is defined (upto a proportionality constant) by

$$f_Y(\varphi_z(\widetilde{y}_{n+1}))J_z(\widetilde{y}_{n+1})f_Z(z). \tag{12}$$

Call the result $z_{n+1}^{(2)}$ and set $y'_{n+1} = \varphi_{z_{n+1}^{(2)}}(\widetilde{y}_{n+1})$.

Step 3. Sample X_{n+1} from $f_{X|Y}(\cdot|y'_{n+1})$

The difference between the standard DA algorithm in (9) and PX-DA is step 2. Per iteration, PX-DA has a slightly greater computational cost due to the need to sample the variables $(Z_n^{(1)}, Z_n^{(2)})$. However, in the cases of practical interest, these variables are typically of a much lower dimension than X or Y and the increase in computational cost is negligible. The benefit though, in terms of the mixing rate of the sampler, has been observed to be quite substantial (Liu et al. 1999). Direct simulation from the probability density on Λ given by (12) is possible for the specific family of mappings $\{\varphi_\lambda\}_{\lambda\in\Lambda}$ we consider in the sequel. When a direct draw from (12) is possible the resulting PX-DA algorithm is termed *exact*.

Step 2 transforms the simulated random variable in step 1 from y_{n+1} to y'_{n+1} via the intermediate value $\widetilde{y}_{n+1}$. Essentially step 2 is implementing a Markov transition from $\mathbb{R}^q$ to $\mathbb{R}^q$ using the kernel

$$Q(y_{n+1}, B) = \mathbb{E}\left\{\mathbb{I}_B(Y'_{n+1})|Y_{n+1} = y_{n+1}\right\} = \mathbb{E}\left\{\mathbb{I}_B(\varphi_{Z_{n+1}^{(2)}} \circ \varphi^{-1}_{Z_{n+1}^{(1)}}(Y_{n+1}))|Y_{n+1} = y_{n+1}\right\}$$

It can be shown that Q is reversible with respect to f_Y and thus f_Y is also invariant for $Q(y_n, B)$ (Liu et al. 1999, Theorem 1). This in turn implies that the invariant probability density of the

Markov chain $\{X_n\}_{n\geq 0}$ generated by the PX-DA algorithm is indeed f_X; if $X_n \sim f_X$ then the law of Y_{n+1} is f_Y and, since f_Y is invariant for Q, the law of Y'_{n+1} is also f_Y.

As was noted by Liu et al. (1999), Meng and van Dyk (1999), it is possible to reduce the auto-correlation between the successive X_n samples generated by the PX-DA algorithm by making the prior f_Z more diffuse. In fact, with a trivial modification, the PX-DA algorithm is still valid when the prior is improper. As the random draw in step 2a is no longer well defined, the correct procedure in this case is to omit this draw and set $\widetilde{y}_{n+1}$ to y_{n+1} from step 1. All other steps remain unchanged. We demonstrate the effect of an improper prior in the numerical examples in Section 5.

The PX-DA algorithm for inferring the value function will be implemented with the following specific transformation of the augmented data. Let $\Lambda = \mathbb{R}_+ \times \mathbb{R}$ and

$$\varphi_z(y) = \frac{y}{z_1} - z_2\mathbf{1}, \qquad z = (z_1, z_2) \in \mathbb{R}_+ \times \mathbb{R}. \tag{13}$$

A result concerning the correctness of the PX-DA method when f_Z is improper is now stated. Although this has been established in the case of either scaling or translation only (Liu et al. 1999; Meng and van Dyk 1999), the extension to the present setting is not difficult.

Proposition 1 *Consider the transformation in (13) and let* $c(y) = \int_{\mathbb{R}_+\times\mathbb{R}} f_Y(\varphi_z(y))J_z(y)dz_1dz_2$, $y \in \mathbb{R}^q$. *Then the Markov transition density on* $\mathbb{R}^q$ *defined by*

$$Q(y, B) = \int_{\mathbb{R}_+\times\mathbb{R}} \mathbb{I}_B(\varphi_z(y)) \frac{f_Y(\varphi_z(y))J_z(y)}{c(y)} dz_1dz_2$$

is reversible with respect to f_Y.

(Proof is in the Appendix.)

For any $h : \mathbb{R}^p \to \mathbb{R}$ which is square-integrable with respect to f_X, i.e. $\int h^2(x)f_X(x)dx < \infty$, if a central limit theorem holds, then

$$\frac{1}{\sqrt{L}} \sum_{n=1}^{L} h(X_n) \xrightarrow{d} \mathcal{N}(\mathbb{E}_{f_X}(h(X)), \sigma^2(h)) \tag{14}$$

where

$$\sigma^2(h) = c_0(h) + 2\sum_{i=1}^{\infty} c_i(h), \qquad c_i(h) = \mathbb{E}_{f_X}(h(X_i)h(X_0)) - \mathbb{E}_{f_X}(h(X_0))^2, \quad i \geq 0. \tag{15}$$

The convergence in (14) is in distribution and the expectations in the expression for $\sigma^2(h)$ are computed with respect to the law of the Markov chain $\{X_n\}_{n\geq 0}$ with initial distribution f_X.

The following inequality for the asymptotic variance of DA, PX-DA (for any proper prior for Z) and PX-DA when the prior for Z is improper holds Hobert and Marchev (2008),

$$\sigma^2(h) \geq \sigma^2_{\text{P}}(h) \geq \sigma^2_{\text{P-I}}(h)$$

where the subscripts indicate the algorithm generating $\{X_n\}_{n\geq 0}$; the standard DA is without subscript, the subscript P denotes PX-DA with a proper prior on Z and P-I denotes PX-DA with an improper prior. PX-DA with an improper prior is said to be the most efficient since it has a smallest asymptotic variance as measured by (14). Strictly speaking, this variance inequality has only been shown to hold when $f_{Y|X}$ can be sampled from exactly in step 1 of Algorithm 1 (Hobert and Marchev 2008). Otherwise, there is no explicit bound quantifying the improvement and we resort to estimating the constants in the expression (15) for the efficiency comparison in the numerical examples.

4 A PX-DA sampler for inferring the value function

The transformation of the augmented data will be as in (13). This section completes the description by specifying the prior for the value function, the auxiliary variable Z and culminates with a statement of the complete sampling algorithm for these specific choices. Extensions to a more general reward function and the practicality of the approach for large problem sizes, specifically large $\mathcal{X}$, are discussed at the end of the section.

Regarding the prior for V, the following requirements seem reasonable: (a) it should respect the symmetry of the model regarding the N states; specifically, it should be invariant with respect to permutation of the state labels; (b) it should be conjugate, so that Gibbs steps can be implemented; and (c) to ease interpretation of the output, it should make the

model identifiable. These requirements are met by the following prior distribution: a Gaussian $\mathcal{N}(\mathbf{0}_N, \kappa I_N)$ distribution, but conditional on the event $\sum_{i=1}^N V(i) = 0$. This prior distribution may be alternately described as follows: take $U \sim \mathcal{N}(\mathbf{0}_N, \kappa I_N)$, then set $V = U - N^{-1}\mathbf{1}_N\mathbf{1}_N^{\mathrm{T}}U$, that is, remove the mean of the $U(i)$ to force the components of V to sum to zero.

The constraint $\sum_{i=1}^N V(i) = 0$ addresses the additive unidentifiability of the model, i.e. the fact that the likelihood is unchanged if the same constant is added to all the $V(i)$. To fix multiplicative unidentifiability, i.e. the likelihood is unchanged if both V and σ are multiplied by the same constant, we take $\Sigma = I$ for the remainder of this Section. This choice presents an important advantage: it makes it possible to implement Step 1 of Algorithm 1 using an efficient Metropolis-Hastings step, as described below. In Section 4.1, we explain briefly how to consider a more general matrix Σ, and why we believe that $\Sigma = I$ should be sufficient in many practical (MDP) applications.

We note in passing a different way to treat additive unidentifiability inspired by multivariate probit models (McCulloch and Rossi 1994): i.e. set one of the N components of the value function to zero, e.g. $V(N) = 0$. In our context however, this would suppress the symmetry between the N states, complicate the notations, and bring no obvious benefit. Also, additive unidentifiability can be exploited to yield a better PX-DA sampler.

The augmented data is

$$Y = (W_1, \ldots, W_T), \text{ with } p(w_1, \ldots, w_T | v) = \prod_{i=1}^T p(w_i | v) = \prod_{i=1}^T \mathcal{N}(w_i; R_i v, I_M)$$

and the PX-DA algorithm defined below will target the joint density ($f_{X,Y}(x, y)$ in section 3).

$$\begin{aligned} &p(v, w_1, \ldots, w_T | d) \\ &\propto \mathcal{N}\left(v; \mathbf{0}_{N-1}, \kappa I_{N-1} - \kappa N^{-1}\mathbf{1}_{N-1}\mathbf{1}_{N-1}^{\mathrm{T}}\right) \prod_{i=1}^T \mathbb{I}_{\{w_i \in \mathbb{R}^M : w_i(a_i) \geq w_i(j), j \neq a_i\}} \mathcal{N}(w_i; R_i v, I_M), \end{aligned} \tag{16}$$

with the slight abuse of notation that the vector v in $\mathcal{N}(w_i; R_i v, I_M)$ is N dimensional where N-th component is

$$v(N) = -\sum_{i=1}^{N-1} v(i).$$

(This convention will hold for the remainder of this section wherever $R_i v$ occurs.) The density (16) has indeed the correct marginal and implementing a Gibbs sampler which samples $(w_1, \ldots, w_T)$ and v alternately is straightforward.

The transformation of the augmented data for the PX-DA implementation is given in (13). We set $f_{Z_1,Z_2}(z_1, z_2) = f_{Z_2}(z_2) f_{Z_1}(z_1)$ and

$$Z_1 \sim IG(a, b), \quad Z_2 \sim \mathcal{N}(0, \kappa/N), \tag{17}$$

where IG is the inverse Gamma density. To clarify the connection with the description of the generic PX-DA sampler in section 3, with a slight abuse of the definition of φ_z^{-1},

$$Y' = \varphi_z^{-1}(Y) = \left(\sqrt{z_1}(W_1^{\mathrm{T}} + z_2 \mathbf{1}_M^{\mathrm{T}}), \ldots, \sqrt{z_1}(W_T^{\mathrm{T}} + z_2 \mathbf{1}_M^{\mathrm{T}})\right)^{\mathrm{T}},$$

and the Jacobian in (11) is

$$J_z(y') = z_1^{-\frac{MT}{2}}.$$

With this choice of transformation of the variables, step 1 and 2a of the generic PX-DA algorithm 1 can be combined into step 1 of algorithm 2 below. Similarly, step 2b and 3 of algorithm 1 may be combined into step 2 of algorithm 2.

The Metropolis Hastings kernel (with independent proposals) for step 1 of Algorithm 2 presented in the Appendix is quite efficient with acceptance rates typically around 70 percent for the numerical examples in Section 5. Step 2 can be implemented as detailed in Section 6.3. When improper priors are used for V, Z_1 and Z_2, the corresponding terms in (20) should be omitted. As discussed in Section 3, when improper priors are used for Z_1 and Z_2, these variables should not be sampled in step 1 above.

4.1 Extensions

4.1.1 State-dependent Rewards

In Section 2 it was assumed that the reward function is not action dependent. The following extension to the criterion in (2) can be considered. Replace $r(X_k)$ in (2) by

$$r(X_k, A_k) = r_1(X_k) + r_2(A_k). \tag{18}$$

Algorithm 2 ***PX-DA for inferring the value function***

Let $w_{1:T}$ and v be the samples after iteration n. At iteration $n+1$, perform the following two steps.

Step 1: Sample $Z_1 \sim IG(a,b)$, call the result z_1, sample $Z_2 \sim \mathcal{N}(0,\kappa/N)$ and let z_2 denote this sampled value. For each $i = 1,...,T$, sample W_i from the truncated Gaussian

$$\mathbb{I}_{\{w_i \in \mathbb{R}^M : w_i(a_i) \geq w_i(j), j \neq a_i\}} \mathcal{N}(w_i; R_i v, I_M), \tag{19}$$

call the result w_i and set $w_i' = \sqrt{z_1}(w_i + z_2 \mathbf{1}_M)$. (This step can be achieved directly or using the Metropolis-Hastings kernel detailed in Section 6.2.)

Step 2: Sample $(V(1),\ldots,V(N-1),Z_2,Z_1)$ from the joint density

$$\mathcal{N}\left(v; \mathbf{0}_{N-1}, \kappa I_{N-1} - \kappa N^{-1} \mathbf{1}_{N-1}\mathbf{1}_{N-1}^{\mathrm{T}}\right) \prod_{i=1}^{T} \mathcal{N}\left(\frac{w_i'}{\sqrt{z_1}} - z_2 \mathbf{1}_M; R_i v, I_M\right)$$
$$\times \mathcal{N}(z_2; 0, \kappa/N) z_1^{-\frac{MT}{2}} IG(z_1; a, b) \tag{20}$$

and $z_1^{-0.5} w_i' - z_2 \mathbf{1}_M$, $i = 1,\ldots,T$, are now the final $w_{1:T}$ for iteration $n+1$.

Note that V^* for this new problem still satisfies (3) with the reward function therein replaced by (18). In this case the action generation model is now

$$A_k = \arg\max_{a \in \mathcal{A}} \{\epsilon_k(a) + r_2(a) + \beta \left(P_a V^*\right)(x_k)\}$$

and

$$p(A_k = i \,|v, r_2, \Sigma, x_k) = \int_{\{\epsilon \in \mathbb{R}^M : \epsilon(i) \geq \epsilon(j), j \neq i\}} \mathcal{N}(\epsilon; r_2 + R_k v, \Sigma) d\epsilon.$$

It can be verified that, for all $(z_1, z_2, z_3) \in \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}$,

$$p(A_k = i \,|\sqrt{z_1}(v + z_2 \mathbf{1}), \sqrt{z_1}(r_2 + z_3 \mathbf{1}), z_1 \Sigma, x_k) = p(A_k = i \,|v, r_2, \Sigma, x_k).$$

The prior for V could be the same as before (see Section 4) and one could also use a prior with the same structure for r_2.

4.1.2 Large State-spaces

Since V is a vector of length $|\mathcal{X}|$, the approach detailed thus far will be impractical for a very large state-space $\mathcal{X}$. In this setting we may regress the value function onto a set of basis functions. (A similar approach was proposed by Geweke and Keane (1996), Geweke and Keane (2000) for a finite horizon dynamic discrete choice problem.) Let $\{\phi_i\}_{1\leq i\leq K}$ be a collection of basis functions, mapping $\mathcal{X}$ to the real line. Typically K is much smaller than $|\mathcal{X}|$. It is assumed that the conditional expectation, $\sum_{x'\in\mathcal{X}}\phi_i(x')p(x'|x,a)$, can be computed easily for each state-action pair (x,a) and i. For example, this would be true if $p(x'|x,a)$ is non-zero for only a handful of values of x', see the human controller example considered in Section 5.2. The action generation model is (for an action independent reward),

$$a_k = \arg\max_{a\in\mathcal{A}}\left\{\epsilon_k(a) + \sum_{i=1}^{K} V^*(i)\left(P_a\phi_i\right)(x_k)\right\},$$

and the corresponding likelihood satisfies

$$p(A_k = i\,|\sqrt{z_1}v, z_1\Sigma, x_k) = p(A_k = i\,|v, \Sigma, x_k).$$

The likelihood is no longer invariant to scalar translations of the value function. As the model is no longer additively unidentifiable, an unconstrained prior may thus be defined over all N components of V. For example, the prior $N(\mathbf{0}_N, \kappa I_N)$ is admissible even as $\kappa\to\infty$. The PX-DA implementation for this model will involve transforming the augmented data by a scalar multiplication only.

4.1.3 Constrained Actions

In some applications, state dependent action constraints are present, i.e. not every action in $\mathcal{A}$ is permitted in every state. The modification to Algorithm 2 is trivial. For example, if action j is not permitted in state x_i, then row j of the R_{x_i} defined in (6) is deleted. Action constraints are present in the example studied in Section 5.

4.1.4 Non-identity Noise Covariance Matrix

We think that restricting the model to an identity covariance matrix for the noise term is very reasonable for the following reasons. First, in the MDP context, one is mostly interested in inferring V as Σ is merely a nuisance parameter. Second, since only one action is observed at a time, it seems hard to estimate correlations between the different components of the noise vector. Third, considering a general Σ means that the dimension of the parameter space becomes $O(M^2)$, and the computational burden $O(M^3)$, as opposed to $O(M)$ for both quantities in the $\Sigma = I$ case. (The computational burden increases also because of the greater difficulty to sample the latent variables W_i, as explained below.) This is clearly impractical when M is large.

However, for the sake of completeness, we now explain how to account for a general covariance matrix Σ. The prior suggested in Imai and van Dyk (2005) may adapted to the present setting. A prior for the covariance matrix Σ subject to the constraint $[\Sigma]_{1,1} = 1$ is constructed by normalizing the samples from an inverse Wishart distribution. Specifically, $\widetilde{\Sigma} \sim \mathcal{IW}(\nu, \widetilde{S})$ and $\Sigma = \widetilde{\Sigma}/[\widetilde{\Sigma}]_{1,1}$. Let $z_1 = [\widetilde{\Sigma}]_{1,1}$ then,

$$p(z_1, \Sigma) \propto |\Sigma|^{-(\nu+M+1)/2} \exp\left(-\frac{\alpha^2}{2z_1}\mathrm{tr}(S\Sigma^{-1})\right)(z_1)^{-\frac{\nu M}{2}-1}$$

where constant α^2 satisfies $\widetilde{S} = \alpha^2 S$. The conditional density

$$p(z_1|\Sigma) = IG\left(\frac{\nu M}{2}, \frac{\alpha^2}{2}\mathrm{tr}(S\Sigma^{-1})\right)$$

is now the new distribution for the scaling parameter in the PX-DA transformation of the augmented data; see (17). To infer Σ as well, Algorithm 2 would be modified to sample $W_{1:T}$, V and then Σ in turn. For a non-diagonal covariance matrix, step 1 cannot be implemented with the Metropolis-Hastings kernel described in Section 6.2. A possible alternative is to use a Gibbs sampling step where, for each i, each component of W_i is sampled conditioned on the remaining components. Once a complete cycle has been performed, then the transformation at the end of step 1 can be applied. Step 2 will be modified to sample (Z_1, Z_2, V, Σ) conditioned on $w_{1:T}$, which can be performed by an appropriate blocking scheme after the change of variable

in (30). Roughly speaking, (Z_1, Z_2, V) is sampled conditioned on $(\Sigma, w_{1:T})$ and then (Z_1, Σ) conditioned on $(Z_2, V, w_{1:T})$. The samples produced may suffer from much more correlation than in the case of Algorithm 2 which is catered to $\Sigma = I$.

5 Numerical Examples

5.1 Toy Example

To demonstrate the performance improvements of PX-DA over standard DA, a data record of 20 state-action pairs was generated from the model with 7 states, 3 actions. The true value function was drawn from the prior. Algorithm 2 was run for 5×10^5 iterations and half were discarded for burn in. The parameters of the priors in (17) were chosen $a = b = 1$, $\kappa = 2500$. Figure 1 shows the computed autocorrelation for some of the components of the estimated value function. The improvements due to scaling and translation of the augmented data are isolated. For the components of the value function not shown, the improvements were comparable. Figure 1 also displays the kernel density estimate of the posterior of component seven of the value function to show that PX-DA indeed preserves the target distribution. The acceptance rate for the Metropolis-Hastings kernel used to implement step 1 of Algorithm 2 was in excess of 95%.

Figure 2 isolates the effect of an improper prior in PX-DA. This study is restricted to PX-DA that scales the augmented data only since the prior for the value function in (16) also depends on the parameter κ that controls the variance of the law of the translation parameter (17). Figure 2 shows the computed autocorrelation for components 4 and 7 of the estimated value function. For the proper prior, $a = 5$, $b = 0.5$. In this case the improper prior for the scaling parameter yields a modest improvement in performance over the proper prior. (Note though there no longer the issue of tuning the prior for the scaling parameter.)

The final experiment demonstrates a situation where PX-DA converges but DA does not. The data comprising of 20 state-action pairs of the previous examples is extended to 50 by appending 30 more. In this example, the priors for V, Z_1 and Z_2 are improper. The

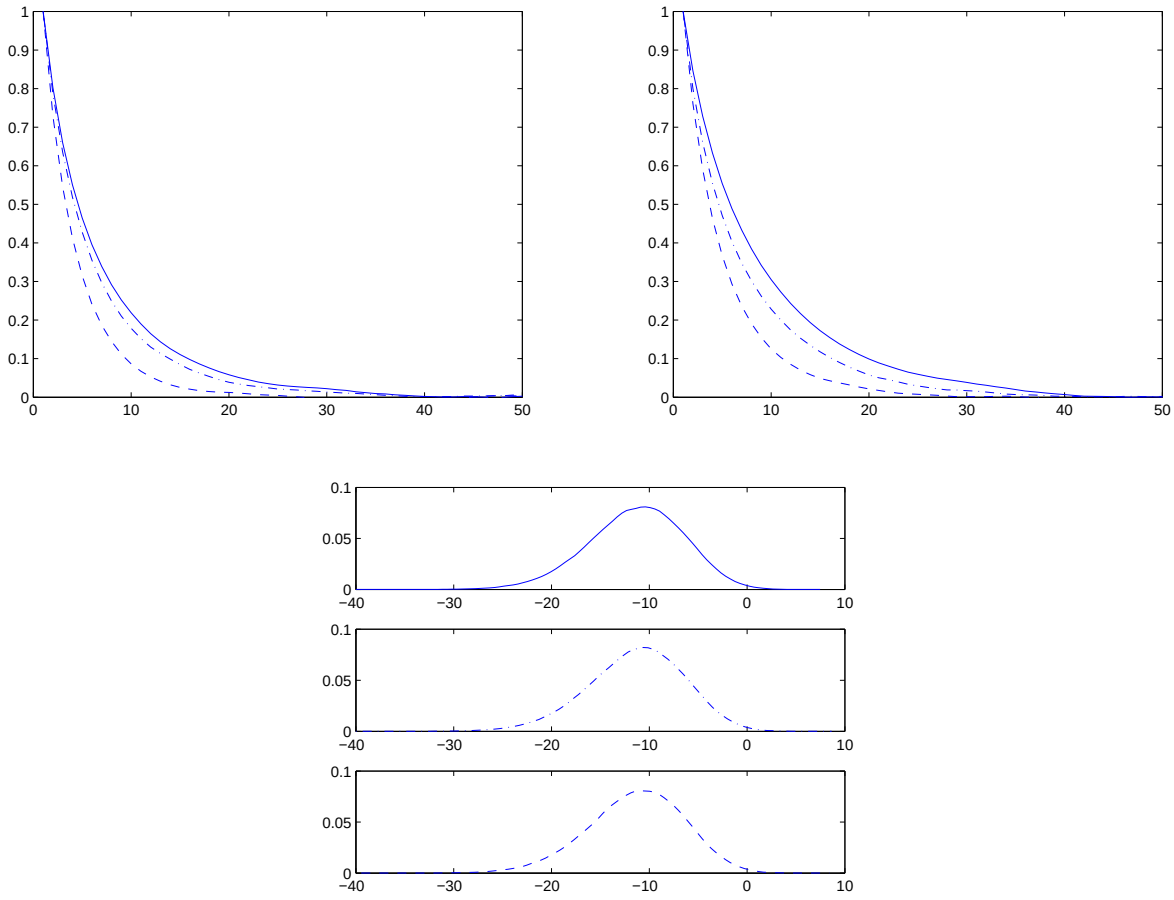

Figure 1: Starting from top left, PX-DA post-burn in autocorrelation plots of posterior samples for component 4 and 7 of the value function; and kernel density estimate of posterior of component 7 of the value function. Solid line is standard DA, dash-dot is PX-DA with scale move only, dashed line is PX-DA with scale and translation move.

posterior mean of the value function calculated with PX-DA with scaling and translation is $[2.34, -0.92, -1.02, 4.12, 5.69, -1.79, -8.41]^{\mathrm{T}}$. ($1.5 \times 10^6$ posterior samples but half discarded for burn in. The posterior mean for PX-DA with scaling or translation only was practically the same.) The PX-DA implementations were initialized with the value function set to $100 \times \mathbf{1}_7$. Shown in Figure 3 is the trace plot of the samples of component 7 of the value function obtained using the DA method initialized with the value function set to $10 \times \mathbf{1}_7$. The mean of the second half of the samples in Figure 3 is -3.56. (In fact all other components of the mean

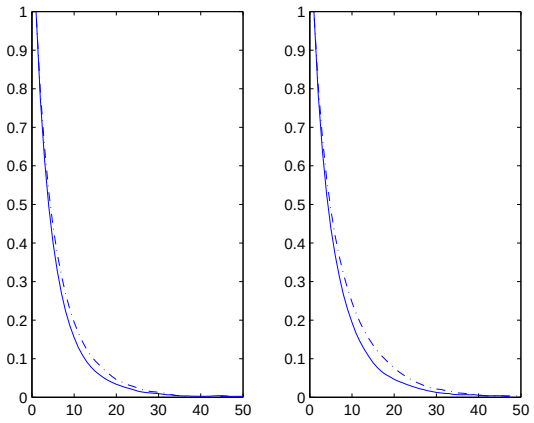

Figure 2: PX-DA post-burn in autocorrelation plots of posterior samples for component 4 (left) and 7 (right) of the value function. Dash-dot line is PX-DA with scale move only and a proper prior with $a = 5$, $b = 0.5$. Solid line is PX-DA with scale move only and an improper prior.

of the posterior value function calculated with DA are quite far out.) In this case we see that DA fails to converge even though initialized far closer to the true values than PX-DA. Finally, to isolate the improvements due to scaling and translation, the autocorrelation plots of certain components of the posterior samples of the value function are compared in Figure 3 for PX-DA implemented with both additive and scaling, scaling only and additive only. In this example, the translation move appears more beneficial than scaling.

5.2 Application to Human Controller Learning

In this section we apply the proposed method to a MDP which arises in the context of the popular computer game Tetris. In this game the player controls the positions and orientations of random two-dimensional shapes, henceforth the *blocks*, which arrive over time and occupy a field of play, henceforth the *board*, in a non-overlapping manner.

5.2.1 Model Definition

In the MDP formulation of Tetris, the state $X = (\zeta, \eta)$ consists of two components. The first component, ζ, is the current configuration of the board and expressed as a 30×10 binary

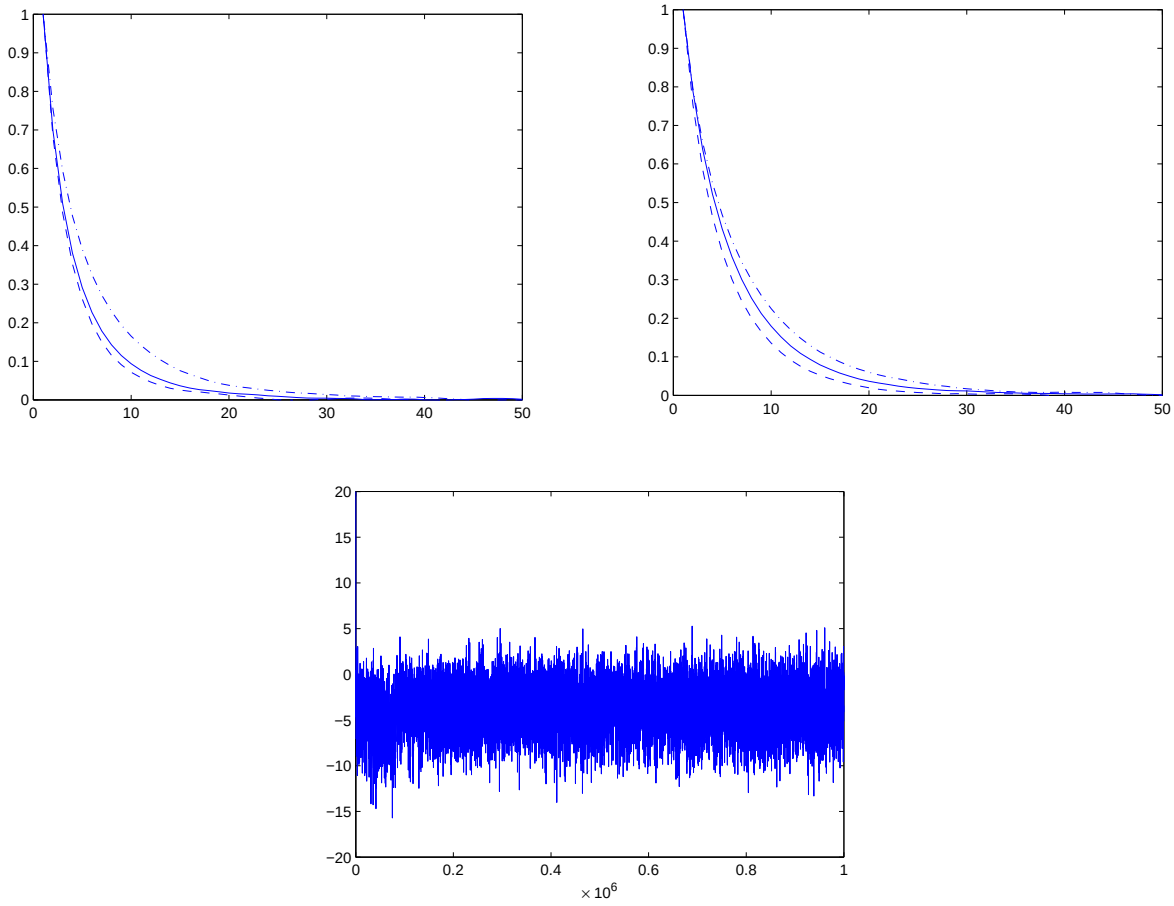

Figure 3: From left to right: PX-DA post-burn in autocorrelation plots of posterior samples for component 4 and 7 of the value function; and DA trace plot of posterior samples of component 7 of the value function. For the autocorrelation plots, solid line is PX-DA with translation move only, dash-dot is PX-DA with scale move only, dashed line is PX-DA with scale and translation move. The mean of the second half of the samples from DA is -3.56 whereas the true posterior mean (calculated with PX-DA for which the three implementations are in agreement) is -8.4.

matrix. The second component, η, is the index of a block. We consider 7 distinct blocks, shown in Figure 4, and thus η takes values in $\{1, 2, ..., 7\}$. Each action A consists of the angle through which to rotate the current block (0°, 90°, 180°, 270°), and the number of squares by which to move it left or right. For each state not all combinations of horizontal translation and rotation are necessarily permitted as the block must remain entirely within the boundaries of the board and must not overlap with any occupied squares. We write $\mathcal{A}(x)$ for the set of

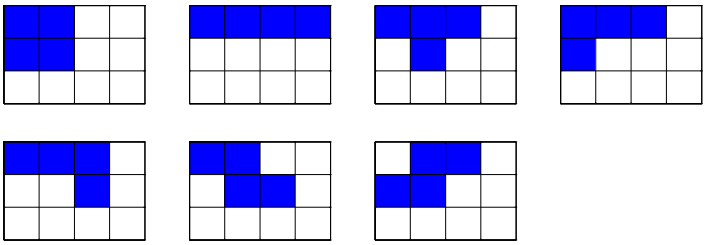

Figure 4: The seven blocks of Tetris.

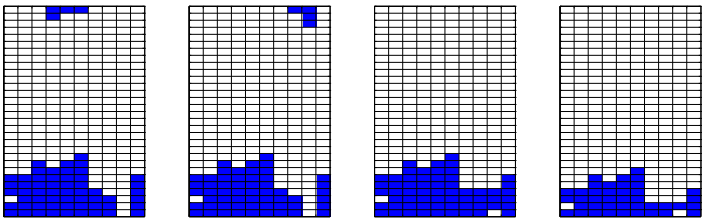

Figure 5: Example iteration of Tetris. From left to right: 1) A block appears at the top of the board. 2) An action is taken to rotate and translate the block. 3) The block then falls until it reaches occupied squares. 4) Fully occupied rows of the board are removed.

actions which are valid in state $X = x$.

From the current state $X_k = (\zeta_k, \eta_k)$ and an action $A_k \in \mathcal{A}(X_k)$, the evolution of the state occurs according to

$$\zeta_{k+1} = \psi(\zeta_k, \eta_k, A_k), \qquad \eta_{k+1} \sim \mathcal{U}(1, 2, ..., 7), \tag{21}$$

where ψ is a deterministic mapping which describes the evolution of the board configuration once the action has been chosen. For a configuration ζ_k with no occupied squares in the top row, ψ yields the new configuration ζ_{k+1} by moving the block η_k according to A_k, then allowing the block to "fall" until it reaches an occupied square or the bottom row of the board, and then removing any fully occupied rows. For a configuration ζ_k which has an occupied square in the top row, ψ sets $\zeta_{k+1} = \zeta_k$ irrespective of A_k and η_k. The latter corresponds to "termination" of the game; once such a state is reached, subsequent actions do not influence the state. A

pictorial representation of one iteration of the game is given in Figure 5.

As each state consists of a single board configuration and block type, the total number of states in the Tetris model is rather large. We therefore adopt the approach outlined in Section 4 and regress the value function on to a collection of K basis functions $\{\phi_1, \phi_2, ..., \phi_K\}$, which depend on the board configuration ζ but not on the randomly falling piece η. (Note that the latter is not controlled as new blocks arrive independently of the action and the previous state.) Specific details of the basis functions are given in section 5.2.

We assume that the reward is independent of the action and the action generation model is then

$$A_k = \arg \max_{a \in \mathcal{A}(x_k)} \left\{ \epsilon_k(a) + \sum_{i=1}^{K} V^*(i) \cdot (\phi_i \circ \psi)(\zeta_k, \eta_k, a) \right\}. \tag{22}$$

We assume that the noise corrupting the action choice has identity covariance. The likelihood of observed data is

$$p(d|v, \Sigma) = \prod_{k=1}^{T} p(a_k|v, x_k, \Sigma),$$

where for each $k = 1, ..., T$,

$$p(A_k = i \,|v, x_k, \Sigma) = \int_{\{\epsilon \in \mathbb{R}^{M_k} : \epsilon(i) \geq \epsilon(j), j \neq i\}} \mathcal{N}(\epsilon; R_k v, I_{M_k}) d\epsilon.$$

Here $M_k := |\mathcal{A}(x_k)|$ and R_k is the $M_k \times N$ matrix with entries specified by

$$[R_k]_{ij} := (\phi_j \circ \psi)(\zeta_k, \eta_k, i).$$

In this case the likelihood is invariant to scaling in the sense that

$$p(d|v, \sigma^2 I) = p(d|\sqrt{z_1}v, z_1\sigma^2 I), \quad \forall z_1 \in \mathbb{R}_+.$$

The sampling algorithm for inference is Algorithm 2 where the augmented data and transformation are given by

$$Y = (W_1, \ldots, W_T), \qquad Y' = \varphi_{z_1}^{-1}(Y) = \left(\sqrt{z_1}W_1^{\mathrm{T}}, \ldots, \sqrt{z_1}W_T^{\mathrm{T}}\right)^{\mathrm{T}},$$

where the scaling factor $Z_1 \sim IG(a, b)$. The Jacobian for this transformation is

$$J_{z_1}(y') = z_1^{-\frac{\sum_{k=1}^{T} M_k}{2}}.$$

The prior for V is $\mathcal{N}(\mathbf{0}_N, \kappa I_N)$.

We consider the following $K = 3$ basis functions which were found to capture various features of the board configuration. ϕ_1, the height of the top-most occupied square in the board, across all the columns; ϕ_2, the number of unoccupied squares which have at least one occupied square above them in the same column; ϕ_3 the sum of the squared differences between occupied heights of adjacent columns.

In Tsitsiklis and Roy (1994), Bertsekas and Tsitsiklis (1996), for a board with c columns, ϕ_1, ϕ_2 and $2c-1$ additional features were used to construct an automated self-improving Tetris playing system using Reinforcement Learning techniques. In contrast, the emphasis here is to make predictions about actions and mimic play on the basis of observed state-action data. In our setup the latter amounts to posterior prediction, which can be performed in the following manner. Let $\{V_n\}_{n=1}^L$ be a collection of post-burn-in samples from the posterior distribution over the value function, obtained from the PX-DA algorithm. Then for each state in a given sequence $\{(\zeta_k, \eta_k)\}_{k=1}^T$ we would like to make predictions under our model about the corresponding action, on the basis of the posterior samples $\{V_n\}_{n=1}^L$. To this end, for each (ζ_k, η_k) we define the MAP predicted action as

$$\begin{aligned} \widehat{A}_k^{\mathrm{MAP}}(\zeta_k, \eta_k) &:= \arg\max_{a \in \mathcal{A}(x_k)} \sum_{n=1}^{L} \mathbb{I}\left[\widehat{A}_{n,k}(\zeta_k, \eta_k) = a\right], \\ \widehat{A}_{n,k}(\zeta_k, \eta_k) &:= \arg\max_{a \in \mathcal{A}(x_k)} \left\{ \epsilon_{n,k}(a) + \sum_{j=1}^{N} V_n(j) \cdot (\phi_j \circ \psi)(\zeta_k, \eta_k, a) \right\}, \end{aligned} \tag{23}$$

where for each $1 \le n \le L$, $1 \le k \le T$ and $a \in \mathcal{A}(x_k)$, $\epsilon_{n,k}(a)$ is an independent $\mathcal{N}(0,1)$ random variable.

In the following section the predictive performance of the model is assessed for a number of data sets. Each data set is divided into two subsets. The PX-DA algorithm is used to draw samples from the posterior corresponding to the first subset and then the accuracy of the posterior prediction is assessed using the second subset. This assessment is performed in terms of the empirical action error, defined as

$$\mathcal{E}_a := \frac{1}{T} \sum_{k=1}^{T} \mathbb{I}\left[\widehat{A}_k^{\mathrm{MAP}}(\zeta_k, \eta_k) \neq a_k\right]. \tag{24}$$

where $\{(\zeta_k, \eta_k, a_k)\}_{k=1}^{T}$ is the second data subset.

Finally we note that in practical situations computation of (23) may be expensive if L is large, in which case one may resort to heuristic action prediction based on a posterior point estimate of V. We do not explore this issue further.

5.2.2 Experiment 1

In the first numerical experiment, we verify that it is possible to recover a value function and perform accurate prediction from data simulated according to the model. We consider three different value functions $(-3, -15, -1)$, $(0, 5, 0)$ and $(-20, 0, 1)$. These value functions were chosen for purposes of exposition; the corresponding optimal policies lead to qualitatively distinct styles of play. Snap-shots of typical board configurations under play according to the action generation model for each of these value functions are given in the top row of Figure 7. The first value function, $(-3, -15, -1)$, led to an "efficient" style of play in which the upper region of the board is rarely occupied. The second value function, (0 5 0), yields a policy which encloses many unoccupied spaces, leading to the distinctive zig–zag pattern displayed in the second columns of Figure 7. The third value function, (−20 0 1), corresponds to a policy which tends to produce "towers" of occupied squares.

For each of the three value functions, 500 observations (state/action pairs) were generated according to the model (22) with the state updated according to (21). During generation of the data, if the game terminated it was immediately restarted. For the value function (−3 −15 −1), termination did not occur within 500 time steps of the game. For the other two, termination typically occurred after 10 to 20 time steps so the full data record of length 500 consisted of the concatenation of several data sets. In all three cases, the first 100 observations were reserved for inference and the remaining 400 used for assessment of predictive performance.

For each value function the PX-DA algorithm, incorporating the Metropolis-Hastings kernel, was run independently targeting the posterior distributions corresponding to the first 10, 20, 50 and 100 observations. In each case the algorithm was run for 5×10^5 iterations, with a burn in of 10^4 iterations. The Metropolis-Hastings acceptance rate was found to be between 0.5 and 0.9 in

all cases. The parameters of the model were set to $\kappa = 2500$ to give a relatively uninformative prior over the value function, and for the prior on the parameter z_1, $a = 3$ and $b = 10^5$. For these tuned values of a and b, using an improper prior over z_1 led to negligible improvements in performance. Post-burn in trace plots, histograms and kernel density estimates are shown in Figure 6 along with the true value function values for the case of inference from 50 observations. In all cases, the posterior marginals have significant mass in the neighborhood of the true value function values.

Figure 6 also shows the autocorrelation for one component of one of the value function, from the output of the PX-DA and standard DA algorithms. This indicates that the PX-DA algorithm yields a significantly lower autocorrelation than the standard DA scheme.

Figure 7 shows the predictive performance in terms of the prediction error $\mathcal{E}_a$ defined in equation (24) as a function of the number of observations used for inference. In all cases $\mathcal{E}_a$ was computed using the remaining 400 observations, i.e. in (24) $T = 400$. These results verify that the predictive performance improves as the number of observations used for inference increases.

The qualitative characteristics of play according to the three true value functions and according to the posterior predictions are also summarized in Figure 7. In this Figure, the top row shows snap-shots of board configurations simulated from the model. The bottom row shows snap-shots of play according to posterior predicted actions (with inference based on 50 observations) for a *different* block sequence $\{\eta_k\}$ and with the state updated according to

$$\zeta_{k+1} = \psi\left(\zeta_k, \eta_k, \widehat{A}_k^{\mathrm{MAP}}(\zeta_k, \eta_k)\right), \qquad \eta_{k+1} \sim \mathcal{U}(1, 2, ..., 7).$$

These results indicate that the predicted actions result in a style of play which is qualitatively similar to that obtained from actions generated according to the true value function.

5.2.3 Experiment 2

In the second experiment, inference was performed from a data set of a human player, i.e. in this case the true value function is unknown. The game was played for 500 iterations and again, the first 100 observations were reserved for inference and the subsequent 400 observations were reserved for assessment of predictive performance. The PX-DA algorithm was run using the

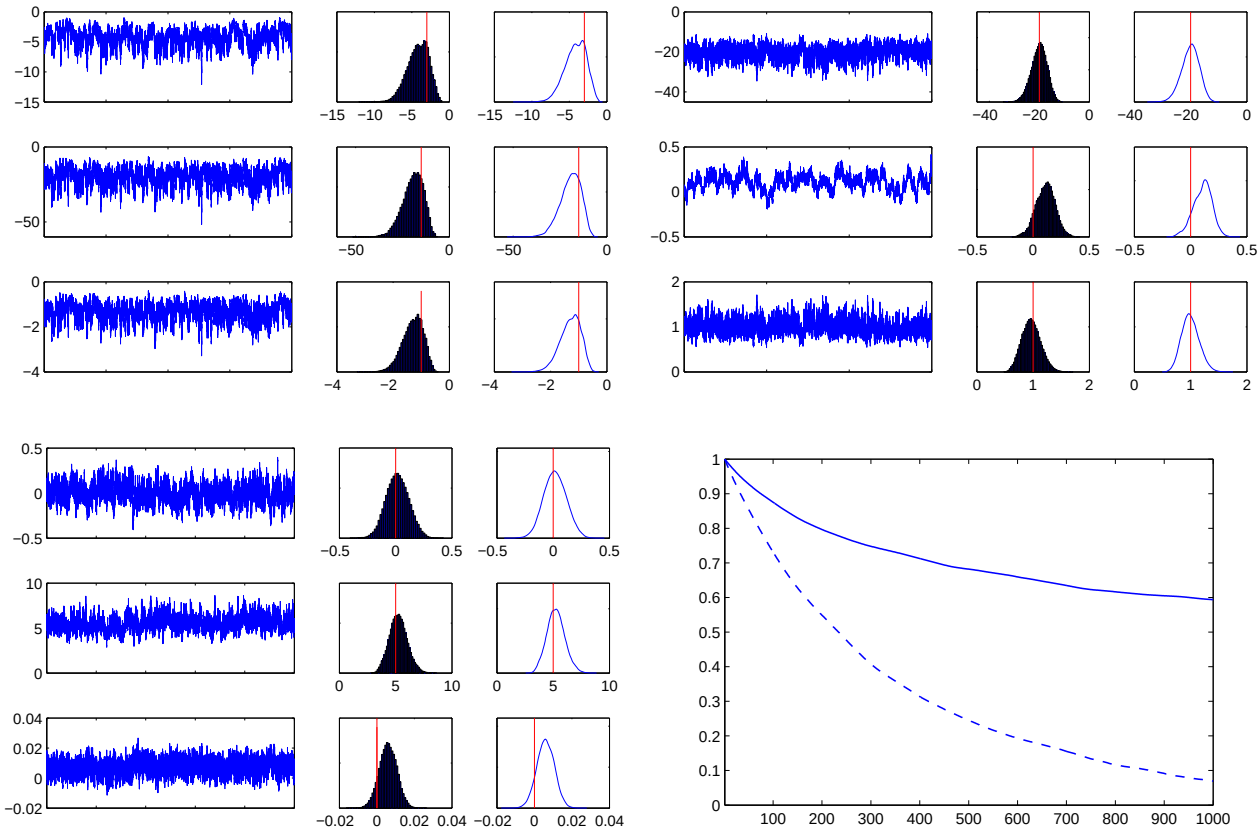

Figure 6: Experiment 1. PX-DA post burn-in trace plots of 4×10^5 samples, histograms and kernel density estimates of the posterior marginal distributions corresponding to 50 observations for the three value functions. Top left: value function $(-3 \ -15 \ -1)$. Top right: $(-20\ 0\ 1)$. Bottom left: $(0\ 5\ 0)$. True values are shown with vertical lines. Bottom right: auto-correlation as a function of lag of the first component of V for PX-DA (dashed) and DA (solid) algorithms in the case of the true value function $(-3 \ -15 \ -1)^T$, from 4×10^5 post burn-in samples.

same settings as in Experiment 1. Again, the Metropolis-Hastings acceptance rate was found to be between 0.5 and 0.9. Trace plots, histograms and kernel density estimates are displayed in Figure 8 for the case of inference from 50 observations. Figure 8 also shows the empirical action error as function of the number of observations used for inference. The result indicates that even with three basis functions, it is possible to capture significant information about the player's policy.

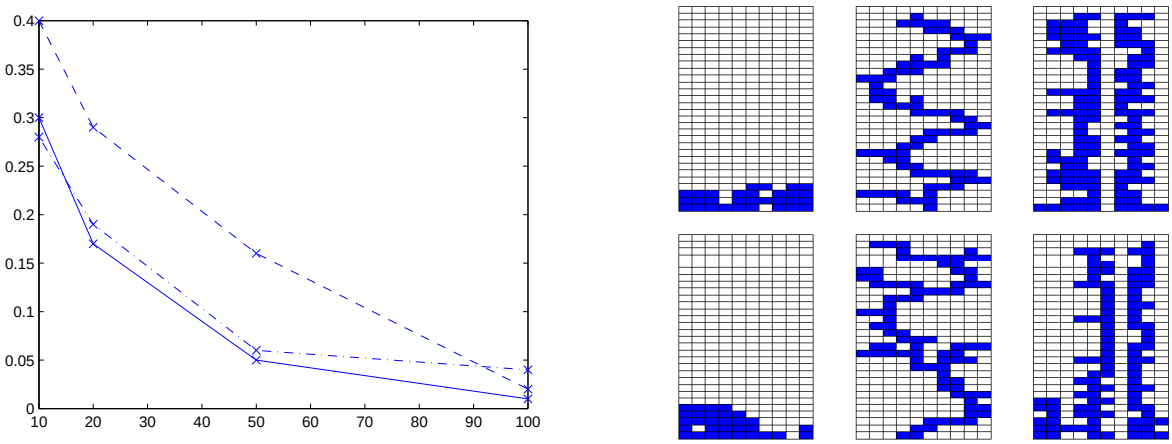

Figure 7: Experiment 1. Right top row: Board snap-shots for simulated data. From left to right the true value functions are $(-3\ -15\ -1)$, $(0\ 5\ 0)$ and $(-20\ 0\ 1)$. Right bottom row: Board snapshots during play according to posterior predictive actions for a different block sequence. Left: Posterior prediction errors as a function of the number of observations used for inference for the three value functions: $(-3\ -15\ -1)$ solid, $(0\ 5\ 0)$ dashed and $(-20\ 0\ 1)$ dash-dot.

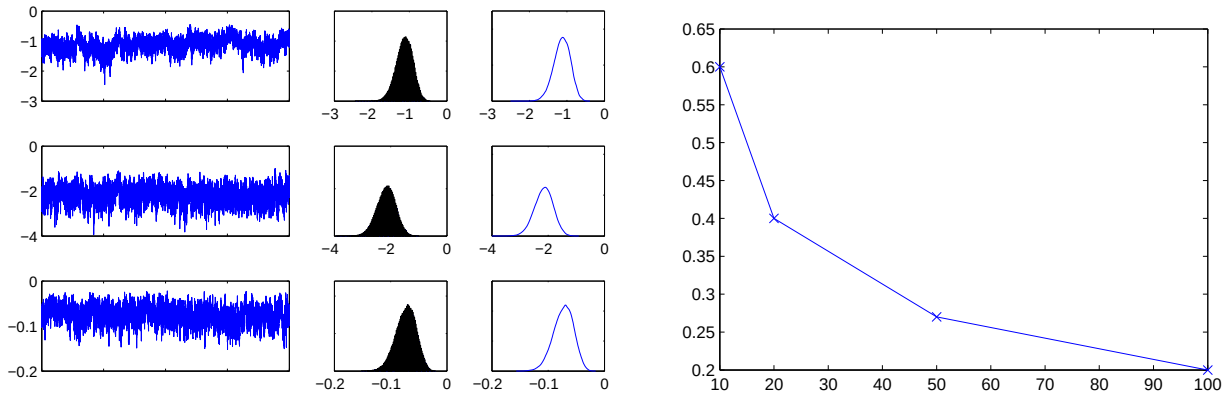

Figure 8: Left: trace plots, histograms and kernel density estimates for of posterior marginals for the three components of the value function. Inference based on 50 observations. Right: Posterior prediction error as a functions of the number of observations used for inference.

6 Appendix

6.1 Proof of Proposition 1

The proof is based on that of (Hobert and Marchev 2008, Proposition 3).

An operation on $\mathbb{R}_+ \times \mathbb{R}$ is defined as follows. For any constants $\tilde{z} = (\tilde{z}_1, \tilde{z}_2), z = (z_1, z_2) \in \mathbb{R}_+ \times \mathbb{R}$, let

$$\tilde{z}z := (\tilde{z}_1 z_1, \tilde{z}_2 + \frac{z_2}{\tilde{z}_1}), \qquad\qquad z^{-1} := (z_1^{-1}, -z_1 z_2).$$

As a consequence of these definitions, $\varphi_{\tilde{z}}(\varphi_z(y)) = \varphi_{\tilde{z}z}(y)$, $\varphi_{z^{-1}}(\varphi_z(y)) = \varphi_z(\varphi_{z^{-1}}(y)) = y$ and $\varphi_z^{-1}(y) = \varphi_{z^{-1}}(y)$. The following equivalences may be established by routine integration. For any $z \in \mathbb{R}_+ \times \mathbb{R}$ and integrable functions $h_1, h_2 : \mathbb{R}^q \to \mathbb{R}^q$, $g : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$,

$$\int h_1(\varphi_z(y)) J_z(y) dy = \int h_1(y) dy, \tag{25}$$

$$\int_{\mathbb{R}_+ \times \mathbb{R}} g(z\tilde{z}) dz_1 dz_2 = \frac{1}{\tilde{z}_1} \int_{\mathbb{R}_+ \times \mathbb{R}} g(z) dz_1 dz_2, \tag{26}$$

$$c(\varphi_z(y)) = \frac{c(y)}{z_1 J_z(y)}, \tag{27}$$

$$\int_{\mathbb{R}_+ \times \mathbb{R}} g(z_1^{-1}, -z_1 z_2) \frac{1}{z_1} dz_1 dz_2 = \int_{\mathbb{R}_+ \times \mathbb{R}} g(z_1, z_2) dz_1 dz_2. \tag{28}$$

(25) is a change of variable formula while (27) follows from (26) and the fact that

$$J_z(\varphi_{\tilde{z}}(y)) = \frac{J_{z\tilde{z}}(y)}{J_{\tilde{z}}(y)}.$$

Then,

$$\begin{aligned}
&\int_{\mathbb{R}_+ \times \mathbb{R}} \int_{\mathbb{R}^q} h_1(y) h_2(\varphi_z(y)) \frac{f_Y(\varphi_z(y)) J_z(y)}{c(y)} f_Y(y) dy dz_1 dz_2 \\
&= \int_{\mathbb{R}_+ \times \mathbb{R}} \left[\int_{\mathbb{R}^q} h_1(\varphi_{z^{-1}}(\varphi_z(y))) h_2(\varphi_z(y)) \frac{f_Y(\varphi_z(y)) J_z(y)}{c(\varphi_{z^{-1}}(\varphi_z(y)))} f_Y(\varphi_{z^{-1}}(\varphi_z(y))) dy \right] dz_1 dz_2 \\
&= \int_{\mathbb{R}_+ \times \mathbb{R}} \left[\int_{\mathbb{R}^q} h_1(\varphi_{z^{-1}}(y)) h_2(y) \frac{f_Y(y)}{c(\varphi_{z^{-1}}(y))} f_Y(\varphi_{z^{-1}}(y)) dy \right] dz_1 dz_2 \\
&= \int_{\mathbb{R}^q} \left[\int_{\mathbb{R}_+ \times \mathbb{R}} h_1(\varphi_{z^{-1}}(y)) h_2(y) \frac{f_Y(y) J_{z^{-1}}(y)}{z_1 c(y)} f_Y(\varphi_{z^{-1}}(y)) dz_1 dz_2 \right] dy \\
&= \int_{\mathbb{R}^q} \left[\int_{\mathbb{R}_+ \times \mathbb{R}} h_1(\varphi_z(y)) h_2(y) \frac{f_Y(y) J_z(y)}{c(y)} f_Y(\varphi_z(y)) dz_1 dz_2 \right] dy
\end{aligned}$$

where the final three lines are established by invoking (25), (27) and (28). This establishes the stated reversibility.

6.2 Metropolis Hastings Kernel

For each $i = 1, ..., T$, sample W_i from the truncated Gaussian given in (19). The procedure for performing this step is discussed below for W_1 (with the subscript omitted from the notation.)

Let $W(i) \sim \mathcal{N}(\mu(i), 1)$ independently, $i = 1, \ldots, M$. The aim is to sample the scalar random variables $W(i)$'s conditional on the event $W(i) < W(l)$, for a fixed l and all $i \neq l$. For convenience, take $l = 1$. The corresponding distribution for the $W(i)$'s may be decomposed as follows. The marginal density of $W(1)$ is

$$p(w(1)) \propto p_u(w(1)) = \mathcal{N}(w(1); \mu(1), 1) \prod_{i=2}^{M} \Phi(w(1) - \mu(i)),$$

and, conditional on $W(1) = w(1)$, $W(i)|\{W(1) = w(1)\} \sim \mathcal{TN}_{(-\infty, w(1)]}(\mu(i), 1)$ for $i = 2, \ldots, M$, independently, where Φ denotes the cumulative distribution function of $\mathcal{N}(0, 1)$, and $\mathcal{TN}_{[a,b]}(m, s^2)$ stands for the $\mathcal{N}(m, s^2)$ distribution truncated to the interval $[a, b]$.

It is trivial to sample the $W(i)$'s, $i > 1$, conditional on $W(1)$ (Devroye 1986, pg. 389), so we focus on the marginal of $W(1)$. We derive an efficient independent Metropolis-Hastings step for $W(1)$ based on a $\mathcal{N}(m, s^2)$ proposal distribution. The acceptance rate reads:

$$\frac{p_u(w'(1))\mathcal{N}(w(1); m, s^2)}{p_u(w(1))\mathcal{N}(w'(1); m, s^2)} \wedge 1$$

where $w(1)$ and $w'(1)$ denote, respectively, the current value and the proposed value $W'(1) \sim \mathcal{N}(m, s^2)$. The main issue is to derive a method for calculating a good Gaussian approximation of $p(w(1))$.

An initial approximation $\mathcal{N}(m_0, s_0^2)$ is first constructed by regarding the function $\Phi(x)$ as the constant one for $x > 0$, and $\mathcal{N}(x; 0, 1)$ for $x < 0$. Specifically, start with $(m_0, s_0^2) = (\mu(1), 1)$, and repeat the following steps: select the factor i with largest $\mu(i)$ and multiply the current Gaussian approximation $\mathcal{N}(x; m_0, s_0^2)$ by either the density $\mathcal{N}(x; \mu(i), 1)$ if $\mu(i) > m_0$, or by 1 otherwise. Discard factor i and repeat this procedure until all $M - 1$ factors have been accounted for. Set (m, s^2) to be the mean and variance of this resulting proposal.

To refine this proposal, perform several Newton-Raphson iterations for finding the mode and the curvature of the mode of $\log p(w(1))$ by using (m, s^2) as the starting values. All these

operations take very little time, and leads to an acceptance rate close to one in most cases. This program is available upon request.

6.3 Implementing Step 2 of Algorithm 2

The density (20) can be written as

$$\mathcal{N}\left(v; \mathbf{0}_{N-1}, \kappa I_{N-1} - \kappa N^{-1} \mathbf{1}_{N-1} \mathbf{1}_{N-1}^{\mathrm{T}}\right) \prod_{i=1}^{T} \mathcal{N}(w_i' - R_i \sqrt{z_1}\,(v + z_2 \mathbf{1}_N)\,; \mathbf{0}_M, z_1 I_M)$$
$$\times \mathcal{N}(z_2; 0, \kappa N^{-1}) IG(z_1; a, b). \tag{29}$$

By implementing the change of variable

$$(v(1), \ldots, v(N-1), z_2)$$
$$\rightarrow (u(1), \ldots, u(N)) = \sqrt{z_1} \left(\left[v(1), \ldots, v(N-1), -\sum_{i=1}^{N-1} v(i) \right] + z_2 \mathbf{1}_N^{\mathrm{T}} \right), \tag{30}$$

(29) becomes

$$\mathcal{N}\left(u; \mathbf{0}_N, \kappa z_1 I_N\right) \prod_{i=1}^{T} \mathcal{N}(w_i' - R_i u; \mathbf{0}_M, z_1 I_M) \times IG(z_1; a, b).$$

Sampling (U, Z_1) is now straightforward and this is to be followed by a transformation of the sampled U variable to recover (V, Z_2). Sample

$$Z_1 \sim IG(\frac{TM}{2} + a, b + SSR/2 + H/2), \quad U|Z_1 = z_1 \sim \mathcal{N}(\frac{1}{z_1} S^{-1} \widetilde{R}^{\mathrm{T}} \widetilde{w}, S^{-1}).$$

where

$$SSR = \widetilde{w}^{\mathrm{T}} \widetilde{w} - \widetilde{w}^{\mathrm{T}} \widetilde{R} (\widetilde{R}^{\mathrm{T}} \widetilde{R})^{-1} \widetilde{R}^{\mathrm{T}} \widetilde{w}, \qquad u_{LS} = (\widetilde{R}^{\mathrm{T}} \widetilde{R})^{-1} \widetilde{R}^{\mathrm{T}} \widetilde{w},$$
$$H = u_{LS}^{\mathrm{T}} (I_N \kappa + (\widetilde{R}^{\mathrm{T}} \widetilde{R})^{-1})^{-1} u_{LS}, \qquad S = I_N z_1^{-1} \kappa^{-1} + z_1^{-1} (\widetilde{R}^{\mathrm{T}} \widetilde{R}).$$

and $\widetilde{w}^{\mathrm{T}} = \left[(w_1')^{\mathrm{T}}, \ldots, (w_T')^{\mathrm{T}}\right]$, $\widetilde{R}^{\mathrm{T}} = \left[R_1^{\mathrm{T}}, \ldots, R_T^{\mathrm{T}}\right]$. Here u_{LS} refers to the least squares estimate of u and SSR is the minimum mean-squared error. Let u denote the sampled random vector U. The sampled (Z_2, V) is $\left(z_1^{-1/2} \frac{\mathbf{1}_N^{\mathrm{T}} u}{N}, z_1^{-1/2} (u - \frac{\mathbf{1}_N^{\mathrm{T}} u}{N} \mathbf{1}_N)\right)$.

References

Abbeel, P. and Ng, A. (2004), "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ed. Brodley, C., Banff, Alberta, Canada: ACM, vol. 69.

Aguirregabiria, V. and Mira, P. (2002), "Swapping the nested fixed point algorithm: a class of estimators for discrete Markov decision models," *Econometrica*, 70, 1519–1543.

Albert, J. and Chib, S. (1993), "Bayesian analysis of binary and polychotomous response data," *Journal of the American Statistical Association*, 88, 669–679.

Bertsekas, D. and Tsitsiklis, J. (eds.) (1996), *Neuro-dynamic programming*, Belmont: Athena Scientific.

Coates, A., Abbeel, P., and Ng, A. Y. (2009), "Apprenticeship learning for helicopter control," *Commun. ACM*, 52, 97–105.

Devroye, L. (ed.) (1986), *Non-uniform random variate generation*, New York: Springer-Verlag.

Geweke, J. and Keane, M. (1996), "Bayesian inference for dynamic discrete choice models without the need for dynamic programming," Working Paper 564, Federal Reserve Bank of Minneapolis.

— (2000), "Bayesian inference for dynamic discrete choice models without the need for dynamic programming," in *Simulation-based inference in econometrics: methods and applications*, eds. Mariano, R., Schuermann, T., and Weeks, M., Cambridge: Cambridge University Press, chap. 4, pp. 100–131.

Geweke, J., Keane, M., and Runkle, D. (1994), "Alternative computational approaches to inference in the multinomial probit model," *Review of Economics and Statistics*, 609–632.

Gotz, G. and McCall, J. (1980), "Estimation in sequential decision making models: a methodological note," *Economic Letters*, 6, 131–136.

Hobert, J. and Marchev, D. (2008), "A theoretical comparison of data augmentation, marginal augmentation and PX-DA algorithms," *The Annals of Statistics*, 36, 532–554.

Hotz, J. and Miller, R. (1993), "Conditional choice probabilities and estimation of dynamic models," *Review of Economic Studies*, 60, 497–529.

Imai, K. and van Dyk, D. (2005), "A Bayesian analysis of the multinomial probit model using marginal data augmentation," *Journal of Econometrics*, 124, 311–334.

Imai, S., Jain, N., and Ching, A. (2009), "Bayesian estimation of dynamic discrete choice models," *Econometrica*, 77, 1865–1899.

Liu, J., Wong, W., and Kong, A. (1999), "Parameter expansion for data augmentation," *Journal of the American Statistical Association*, 94, 1264–1274.

McCulloch, R., Polson, N., and Rossi, P. (2000), "A Bayesian analysis of the multinomial probit model with fully identified parameters," *Journal of Econometrics*, 99, 172–193.

McCulloch, R. and Rossi, P. (1994), "An exact likelihood analysis of the multinomial probit model," *Journal of Econometrics*, 64, 207–240.

Meng, X.-L. and van Dyk, D. (1999), "Seeking efficient data augmentation schemes via conditional and marginal augmentation," *Biometrika*, 86, 301–320.

Ng, A. and Russell, S. (2000), "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ed. Langley, P., San Franciso, CA: Morgan Kaufmann, pp. 663–670.

Nobile, A. (1998), "A hybrid Markov chain for Bayesian analysis of the multinomial probit model," *Statistics and Computing*, 8, 229–242.

Rust, J. (1987), "Optimal replacement of GMC bus engines: an empirical model of Harold Zurcher," *Econometrica*, 55, 999–1033.

— (1988), "Maximum likelihood estimation of discrete control processes," *SIAM Jour. Control and Optim.*, 26, 1006–1024.

Schmajuk, N. A. and Zanutto, B. S. (1997), "Escape, Avoidance, and Imitation: A Neural Network Approach," *Adaptive Behavior*, 6, 63–129.

Tanner, M. and Wong, W. (1987), "The calculation of posterior distributions by data augmentation (with discussion)," *Journal of the American Statistical Association*, 82, 528–550.

Tsitsiklis, J. and Roy, B. V. (1994), "Feature-based methods for large scale dynamic programming," Technical Report LIDS-P 2277, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems.

Watkins, C. (1987), "Learning from delayed rewards," Ph.D. thesis, University of Cambridge.

Wolpin, K. (1984), "An estimable dynamic stochastic model of fertility and child mortality," *Journal of Political Economy*, 92, 852–874.