Appendices for "Action-Sufficient State Representations for Policy Learning"

Anonymous Author(s)
Affiliation
Address
email

A An Example of d-Separation

We give an example to illustrate d-separation. In Figure 9, x_1 and x_5 are d-separated, given any subset of $\{x_3, x_4\}$, but they are not d-separated given $\{x_2\}$, $\{x_6\}$, or $\{x_2, x_6\}$.

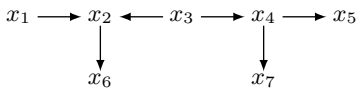

Figure 9: A graph illustrating d-separation.

B Proof of Proposition 1

Proof. We first show that, among the components of $\vec{s}_t$, $\vec{s}_t^{\text{ASR}}$ contains sufficient and necessary information for policy learning $\Longrightarrow$ any $s_{i,t} \in \vec{s}_t^{\text{ASR}}$ satisfies at least one of the three conditions in Proposition 1.

$\vec{s}_t^{\text{ASR}}$ containing sufficient and necessary information for policy learning among the components of $\vec{s}_t$ implies that

- $\forall s_{i,t} \in \vec{s}_t^{\text{ASR}}, \forall \tilde{s}_t \in \mathcal{P}(\vec{s}_{-i,t}), s_{i,t} \not\perp\!\!\!\perp a_t | \{\tilde{s}_t, R^c\}$, and
- $\forall s_{i,t} \notin \vec{s}_t^{\text{ASR}}, s_{i,t} \perp\!\!\!\perp a_t | \{\vec{s}_t^{\text{ASR}}, R^c\}$,

where $\not\perp\!\!\!\perp$ denotes probabilistically dependence and $\perp\!\!\!\perp$ denotes probabilistically independence, and $\vec{s}_{-i,t}$ denotes remaining dimensions of $\vec{s}_t$ after removing $s_{i,t}$ and $\mathcal{P}$ is its power set.

According to the global Markov condition, if $s_{i,t} \not\perp\!\!\!\perp a_t | \{\tilde{s}_t, R^c\}$, then a_t and $s_{i,t}$ are d-connected. Thus, according to the definition of d-separation, any $s_{i,t} \in \vec{s}_t^{\text{ASR}}$ satisfies one of the three conditions in Proposition 1.

Furthermore, according to the faithfulness assumption, if $s_{i,t} \perp\!\!\!\perp a_t | \{\vec{s}_t^{\text{ASR}}, R^c\}$, then a_t and $s_{i,t}$ are d-separated, given the collider R^c and $\vec{s}_t^{\text{ASR}}$. Thus, according to d-separation, any $s_{i,t} \notin \vec{s}_t^{\text{ASR}}$ does not satisfy any of the three conditions.

Next, we show that $\forall s_{i,t} \in \vec{s}_t^{\text{ASR}}$ satisfies one of the three conditions $\Longrightarrow \vec{s}_t^{\text{ASR}}$ contains sufficient and necessary information for policy learning among the components of $\vec{s}_t$.

According to d-separation, if $s_{i,t} \in \vec{s}_t^{\text{ASR}}$ satisfies one of the three conditions, then a_t and $s_{i,t}$ are d-connected. Further with the faithfulness assumption, $\forall s_{i,t} \in \vec{s}_t^{\text{ASR}}, s_{i,t} \not\perp\!\!\!\perp a_t | \{\tilde{s}_t, R^c\}$. Thus, $\vec{s}_t^{\text{ASR}}$ contains sufficient information, among the components of $\vec{s}_t$, for policy learning.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Moreover, for any $s_{i,t}$, which does not satisfy any of the three conditions, a_t and $s_{i,t}$ are d-separated, given the collider R^c and $\vec{s}_t^{\text{ASR}}$. Further with the global Markov condition, $\forall s_{i,t} \notin \vec{s}_t^{\text{ASR}}$, $s_{i,t} \perp\!\!\!\perp a_t | \{\vec{s}_t^{\text{ASR}}, R^c\}$. Thus, $\vec{s}_t^{\text{ASR}}$ contains necessary information, among the components of $\vec{s}_t$, for policy learning.

Therefore, $\vec{s}_t^{\text{ASR}}$ contains sufficient and necessary information, among the components of $\vec{s}_t$, for policy learning, if and only if any $s_{i,t} \in \vec{s}_t^{\text{ASR}}$ satisfies one of the three conditions in Proposition 1.

□

C Proof of Proposition 2

Proof. We first show that the framework, given in Eq. (2), describes the generating process in scenarios where the reward is imperfect. Let us first look at the linear case as a specific example, where $r_t = r_t^{\text{p}} + r_t^{\text{ip}}$. In the linear case, we have

$$\begin{cases} r_t^{\text{p}} &= B^1 \vec{s}_{t-1}^{\text{p}} + C a_{t-1} + \epsilon_t^1, \\ r_t^{\text{ip}} &= B^2 \vec{s}_{t-1}^{\text{ip}} + \epsilon_t^2, \end{cases}$$

where B^1, B^2, and C are corresponding coefficients. Hence, $r_t = r_t^{\text{p}} + r_t^{\text{ip}} = (B^1 \vec{s}_{t-1}^{\text{p}} + C a_{t-1} + \epsilon_t^1) + (B^2 \vec{s}_{t-1}^{\text{ip}} + \epsilon_t^2) = (B^1, B^2)\binom{\vec{s}_{t-1}^{\text{p}}}{\vec{s}_{t-1}^{\text{ip}}} + C a_{t-1} + (\epsilon_t^1 + \epsilon_t^2)$. Let $B = (B^1, B^2)$ and $\epsilon_t = \epsilon_t^1 + \epsilon_t^2$, so $r_t = B\vec{s}_{t-1} + C a_{t-1} + \epsilon_t$. That is, r_t is a function of $\vec{s}_{t-1}$ and a_{t-1}, and is disturbed by i.i.d. noise. Thus, it is easy to see that Eq. (2) describes the generating process of the linear case with imperfect reward.

Generally, we represent the reward r_t as an arbitrary function of r_t^{p} and r_t^{ip}, i.e., $r_t = g^0(r_t^{\text{p}}, r_t^{\text{ip}})$, with

$$\begin{cases} r_t^{\text{p}} &= g^1(B^1 \vec{s}_{t-1}^{\text{p}}, a_{t-1}, \epsilon_t^1), \\ r_t^{\text{ip}} &= g^2(B^2 \vec{s}_{t-1}^{\text{ip}}, \epsilon_t^2). \end{cases}$$

Hence,

$$\begin{aligned} r_t &= g^0(r_t^{\text{p}}, r_t^{\text{ip}}) \\ &= g^0\big(g^1(B^1 \vec{s}_{t-1}^{\text{p}}, a_{t-1}, \epsilon_t^1), g^2(B^2 \vec{s}_{t-1}^{\text{ip}}, \epsilon_t^2)\big), \end{aligned}$$

that is, r_t is a function of $\vec{s}_{t-1}^{\text{p}}$, $\vec{s}_{t-1}^{\text{ip}}$, and a_{t-1}, and we can formulate $r_t = g(B\vec{s}_{t-1}, a_{t-1}, \epsilon_t)$. Thus, the generating process with imperfect reward is described by the framework given in Eq. (2).

Next, we show that the framework, given in Eq. (2), also describes the generating process in scenarios where only partial information of ASRs is contained in the perceived signal. We denote the states which have edges to o_t by $\vec{s}_t^{\text{c}}$, and otherwise by $\vec{s}_t^{\text{ic}}$. In this case, the observation-state mapping and the reward function can be represented as follows:

$$\begin{cases} o_t = f(A^1 \vec{s}_t^{\text{c}}, e_t), \\ r_t = g(B\binom{\vec{s}_{t-1}^{\text{c}}}{\vec{s}_{t-1}^{\text{ic}}}, a_{t-1}, \epsilon_t). \end{cases}$$

Interestingly, we can see that although $\vec{s}_t^{\text{ic}}$ is not included in the function of o_t, it is included in the function of r_t. It is easy to re-organize the above equation to fit to our framework given in Eq. (2), by setting $A = [A^1 \quad \mathbf{0}]$. Thus, $o_t = f(A^1 \vec{s}_t^{\text{c}}, e_t) = f([A^1 \quad \mathbf{0}]\binom{\vec{s}_t^{\text{c}}}{\vec{s}_t^{\text{ic}}}, e_t) = f(A\vec{s}_t, e_t)$, which is described by our framework.

Moreover, it is obvious that the framework, given in Eq. (2), describes the generating process in scenarios where the reward is perfect and the perceived signal o_t contains all the information of ASRs.

□

D Assumptions of Theorem 1

To show the identifiability of the model in Eq. (3), we make the following assumptions:

A1. $d_o + d_r \geq d_s$, where $|o_t| = d_o$, $|r_t| = d_r$, and $|s_t| = d_s$.
A2. $(A^\top, B^\top)^\top$ is full column rank and D is full rank.
A3. The control signal a_t is i.i.d. and the state $\vec{s}_t$ is stationary.
A4. The process noise has a unit variance, i.e., $\text{var}(\eta_t) = I$.

E Proof of Theorem 1

Proof. The proof of the linear case without control signals has been shown in [1]. Below, we give the identifiability proof in the linear-Gaussian case with control signals:

$$\begin{cases} o_t = A\vec{s}_t + e_t, \\ r_{t+1} = B\vec{s}_t + Ca_t + \epsilon_{t+1}, \\ \vec{s}_t = D\vec{s}_{t-1} + Ea_{t-1} + \eta_t. \end{cases} \tag{5}$$

Let $\mathbf{y}_{t+1} = [o_t^\top, r_{t+1}^\top]^\top$, $\ddot{A} = [A^\top, B^\top]^\top$, $\ddot{C} = [\vec{0}^\top, C^\top]^\top$, and $\ddot{e}_t = [e_t^\top, \epsilon_{t+1}^\top]^\top$. Then the above equation can be represented as:

$$\begin{cases} \mathbf{y}_t = \ddot{A}\vec{s}_t + \ddot{C}a_t + \ddot{e}_t, \\ \vec{s}_t = D\vec{s}_{t-1} + Ea_{t-1} + \eta_t. \end{cases} \tag{6}$$

Because the dynamic system is linear and Gaussian, we make use of the second-order statistics of the observed data to show the identifiability. We first consider the cross-covariance between $\mathbf{y}_{t+k}$ and a_t:

$$\begin{cases} \text{Cov}(\mathbf{y}_{t+k}, a_t) = \ddot{A}D^{k-1}E \cdot \text{Var}(a_t), & \text{if } k > 0. \\ \text{Cov}(\mathbf{y}_{t+k}, a_t) = \ddot{C} \cdot \text{Var}(a_t), & \text{if } k = 0. \end{cases} \tag{7}$$

Thus, from the cross-covariance between $\mathbf{y}_{t+k}$ and a_t, we can identify $\ddot{A}E$, $\ddot{C}$, and $\ddot{A}D^kE$ for $k > 0$.

Next, we consider the auto-covariance function of $\vec{s}$. Define the auto-covariance function of $\vec{s}$ at lag k as $\mathbf{R}_{\vec{s}}(k) = \mathbb{E}[\vec{s}_t\vec{s}_{t+k}^\top]$, and similarly for $\mathbf{R}_{\mathbf{y}}(k)$. Clearly $\mathbf{R}_{\vec{s}}(-k) = \mathbf{R}_{\vec{s}}(k)^\top$ and $\mathbf{R}_{\mathbf{y}}(-k) = \mathbf{R}_{\mathbf{y}}(k)^\top$. We have

$$\begin{cases} \mathbf{R}_{\vec{s}}(k) = \mathbf{R}_{\vec{s}}(k-1) \cdot D^\top, & \text{if } k > 0, \\ \mathbf{R}_{\vec{s}}(k) = \mathbf{R}_{\vec{s}}^\top(1) \cdot D^\top + E\text{Var}(a_{t-1})E^\top + I, & \text{if } k = 0. \end{cases} \tag{8}$$

Below, we first consider the case where $d_o + d_r = d_s$. Let $\tilde{\mathbf{y}}_t = \ddot{A}\vec{s}_t$, so $\mathbf{y}_t = \tilde{\mathbf{y}}_t + \ddot{C}a_{t-1} + \ddot{e}_t$ and $\mathbf{R}_{\tilde{\mathbf{y}}}(k) = \ddot{A}\mathbf{R}_{\vec{s}_t}(k)\ddot{A}^\top$. $\mathbf{R}_{\tilde{\mathbf{y}}}(k)$ satisfies the recursive property:

$$\begin{cases} \mathbf{R}_{\tilde{\mathbf{y}}}(k) = \mathbf{R}_{\tilde{\mathbf{y}}}(k-1) \cdot \Omega^\top, & \text{if } k > 0, \\ \mathbf{R}_{\tilde{\mathbf{y}}}(k) = \mathbf{R}_{\tilde{\mathbf{y}}}^\top(1) \cdot \Omega^\top + \ddot{A}(E\text{Var}(a_{t-1})E^\top + I)\ddot{A}^\top, & \text{if } k = 0, \end{cases} \tag{9}$$

where $\Omega = \ddot{A}D\ddot{A}^{-1}$.

Denote $S_k = \ddot{A}D^{k-1}E \cdot \text{Var}(a_t)$. Then we can derive the recursive property for $\mathbf{R}_{\mathbf{y}}(k)$:

$$\begin{cases} \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}(k-1) \cdot \Omega^\top - \ddot{C}S_{k-1}^\top\Omega^\top + \ddot{C}S_k^\top, & \text{if } k > 1, \\ \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}(k-1) \cdot \Omega^\top - \ddot{C}\text{Var}^\top(a_t)\ddot{C}^\top\Omega^\top - \Sigma_e\Omega^\top + \ddot{C}S_k^\top, & \text{if } k = 1, \\ \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}^\top(1) \cdot \Omega^\top + \left(\ddot{C}\text{Var}(a_t)\ddot{C}^\top + \Sigma_e\right) + \ddot{A}(E\text{Var}(a_t)E^\top + I)\ddot{A}^\top, & \text{if } k = 0. \end{cases} \tag{10}$$

When $k = 2$, we have

$$\mathbf{R}_{\mathbf{y}}(2) = \mathbf{R}_{\mathbf{y}}(1) \cdot \Omega^\top - \ddot{C}S_1^\top\Omega^\top + \ddot{C}S_2^\top.$$

The above equation can be re-organized as

$$\left(\mathbf{R}_{\mathbf{y}}(2) - \ddot{C} \cdot S_2^\top\right) = \left(\mathbf{R}_{\mathbf{y}}(1) - \ddot{C} \cdot S_1^\top\right) \cdot \Omega^\top.$$

Because $\ddot{C}$ and S_k are identifiable, and suppose $\left(\mathbf{R}_{\mathbf{y}}(1) - \ddot{C} \cdot S_1^\top\right)$ is invertible, $\Omega = \ddot{A}D\ddot{A}^{-1}$ is identifiable.

We further consider $\mathbf{R}_{\mathbf{y}}(0)$ and $\mathbf{R}_{\mathbf{y}}(1)$ and write down the following form:

$$\begin{bmatrix} \mathbf{R}_{\mathbf{y}}(0) - \ddot{A}(E\text{Var}(a_{t-1})E^\top + I)\ddot{A}^\top \\ \mathbf{R}_{\mathbf{y}}(1) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathbf{y}}^\top(1) \\ \mathbf{R}_{\mathbf{y}}(0) \end{bmatrix} \cdot \Omega^\top + \begin{bmatrix} \ddot{C}\text{Var}(a_t)\ddot{C}^\top \\ -\ddot{C}\text{Var}^\top(a_t)\ddot{C}^\top\Omega^\top + \ddot{C}S_1^\top \end{bmatrix} + \Sigma_e \begin{bmatrix} I \\ -\Omega^\top \end{bmatrix}. \tag{11}$$

From the above two equations we can then identify Σ_e and $\ddot{A}(E\text{Var}(a_{t-1})E^\top + I)\ddot{A}^\top$, and because $\ddot{A}E$ is identifiable, $\ddot{A}\ddot{A}^\top$ is identifiable.

In summary, we have shown the identifiability of $\ddot{C}$, $\ddot{A}E$, $\ddot{A}D^kE$, $\ddot{A}\ddot{A}^\top$, and Σ_e. Furthermore, $\ddot{A}$, D, and E are identified up to some orthogonal transformations. That is, suppose the model in Eq. (3) with parameters $(A, B, C, D, E, \Sigma_e, \Sigma_\epsilon)$ and that with $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}, \tilde{E}, \tilde{\Sigma}_{\tilde{e}}, \tilde{\Sigma}_{\tilde{\epsilon}})$ are observationally equivalent, we then have $\tilde{\ddot{A}} = \ddot{A}U$, $\tilde{C} = C$, $\tilde{D} = U^\top DU$, $\tilde{E} = EU$, $\tilde{\Sigma}_{\tilde{e}} = \Sigma_e$, and $\tilde{\Sigma}_{\tilde{\epsilon}} = \Sigma_\epsilon$, where U is an orthogonal matrix.

Next, we extend the above results to the case where $d_o + d_r > d_s$. Let $\ddot{A}_i^\top$ be the i-th row of $\ddot{A}$. Recall that $\ddot{A}$ is of full column rank. Then for any i, one can show that there always exist $d_s - 1$ rows of $\ddot{A}$, such that they, together with $\ddot{A}_i^\top$, form a $d_s \times d_s$ full-rank matrix, denoted by $\bar{\ddot{A}}_i$. Then from the observed data corresponding to $\bar{\ddot{A}}_i$, $\bar{\ddot{A}}_i$ is determined up to orthogonal transformations. Thus, $\ddot{A}$ is identified up to orthogonal transformations. Similarly, $\ddot{C}$, D, and E are identified up to orthogonal transformations. Furthermore, $\text{Cov}(\ddot{A}\vec{s}_t + \ddot{C}a_t)$ is determined by $\ddot{A}$, $\ddot{C}$, D, and E. Because $\text{Cov}(\mathbf{y}_t) = \text{Cov}(\ddot{A}\vec{s}_t + \ddot{C}a_t) + \Sigma_{\ddot{e}}$, $\Sigma_{\ddot{e}}$ is identifiable. □

F Estimation of Linear-Gaussian Models

We extend the estimation without control signals [2] to the case with control signals. We use an expectation-maximization (EM) algorithm for estimation, which iterates between two steps, expectation (E) and maximization (M). In particular, the E step estimates the expected log-likelihood $\mathcal{Q}$, with

$$\mathcal{Q} = \mathbb{E}_{q(\vec{s}_{1:T}|\mathbf{y}_{1:T}, a_{1:T})}\big[\log P(\mathbf{y}_{1:T}, \vec{s}_{1:T})\big] + (\lambda_1\|A\|_1 + \lambda_2\|B\|_1 + \lambda_3\|D\|_1 + \lambda_4\|E\|_1), \quad (12)$$

where $\mathbf{y}_{1:T} = \{(o_t^\top, r_{t+1}^\top)^\top\}_{t=1}^T$, $a_{1:T} = \{a_t\}_{t=1}^T$, $\vec{s}_{1:T} = \{\vec{s}_t\}_{t=1}^T$, and $\|\cdot\|_1$ denotes the L_1 norm to control the sparsity. The quantity $\mathcal{Q}$ depends on three expectations: $\mathbb{E}_q[\vec{s}_t]$, $\mathbb{E}_q[\vec{s}_t\vec{s}_t^\top]$, and $\mathbb{E}_q[\vec{s}_t\vec{s}_{t-1}^\top]$, which will be denoted by the following symbols:

$$\begin{array}{lcl} \hat{\vec{s}}_t & \equiv & \mathbb{E}_q[\vec{s}_t], \\ P_t & \equiv & \mathbb{E}_q[\vec{s}_t\vec{s}_t^\top], \\ P_{t,t-1} & \equiv & \mathbb{E}_q[\vec{s}_t\vec{s}_{t-1}^\top]. \end{array}$$

We leverage Kalman filter and Rauch-Tung-Striebel smoother for inference.

Let $\ddot{e}_t \sim \mathcal{N}(0, R)$ and $\eta_t \sim \mathcal{N}(0, Q)$. In the M step, we update the parameters by taking the corresponding partial derivative of the expected log-likelihood, setting it to zero and solving. The results are as follows.

$$\begin{array}{lcl}
\ddot{A}^{\text{new}} & = & \big(\sum_{t=1}^T(\mathbf{y}_t - \ddot{C}a_{t-1})\cdot\hat{\vec{s}}_t^\top + \frac{\log T}{2}\lambda_A\cdot R\cdot v_A\big)\cdot(\sum_{t=1}^T P_t)^{-1}, \\
\ddot{C}^{\text{new}} & = & \big(\sum_{t=1}^T(\mathbf{y}_t - \ddot{A}\hat{\vec{s}}_t)\cdot a_{t-1}^\top\big)\cdot(\sum_{t=1}^T a_{t-1}a_{t-1}^\top)^{-1}, \\
R^{\text{new}} & = & \frac{1}{T}\sum_{t=1}^T\big((\mathbf{y}_t - \ddot{C}a_{t-1})(\mathbf{y}_t - \ddot{C}a_{t-1})^\top - 2\ddot{A}\hat{\vec{s}}_t(\mathbf{y}_t - \ddot{C}a_{t-1})^\top + \ddot{A}P_t\ddot{A}^\top\big), \\
D^{\text{new}} & = & \big(\sum_{t=2}^T(P_{t,t-1} - Ea_{t-1}\hat{\vec{s}}_{t-1}^\top + \frac{\log T}{2}\lambda_D\cdot Q\cdot v_D)\big)\cdot(\sum_{t=2}^T P_{t-1})^{-1}, \\
E^{\text{new}} & = & \big(\sum_{t=2}^T(\hat{\vec{s}}_t^\top - D\hat{\vec{s}}_{t-1}^\top)\cdot a_{t-1}^\top + \frac{\log T}{2}\lambda_E\cdot Q\cdot v_E\big)\cdot(\sum_{t=2}^T a_{t-1}a_{t-1}^\top)^{-1}, \\
Q^{\text{new}} & = & \frac{1}{T-1}\sum_{t=2}^T\big(P_t - 2P_{t,t-1}D^\top + DP_{t-1}D^\top - 2(\hat{\vec{s}}_t^\top - D\hat{\vec{s}}_{t-1}^\top)\cdot(Ea_{t-1})^\top + (Ea_{t-1})(Ea_{t-1})^\top\big),
\end{array}$$

where $v_A = \text{sign}(A)$, $v_B = \text{sign}(B)$, $v_D = \text{sign}(D)$, and $v_E = \text{sign}(E)$.

Denote $\mathbb{E}[\vec{s}_t|\mathbf{y}_{1:\tau}]$ by $\vec{s}_t^{\tau}$, and denote $\text{Var}[\vec{s}_t|\mathbf{y}_{1:\tau}]$ by V_t^{τ}. We obtain the following Kalman filter forward recursions:

$$\begin{array}{lcl} \vec{s}_t^{t-1} & = & D\vec{s}_{t-1}^{t-1}, \\ V_t^{t-1} & = & DV_{t-1}^{t-1}D^\top + Q, \\ K_t & = & V_t^{t-1}\ddot{A}^\top(\ddot{A}V_t^{t-1}\ddot{A}^\top + R)^{-1}, \\ \vec{s}_t^{t} & = & \vec{s}_t^{t-1} + K_t(\mathbf{y}_t - \ddot{A}\vec{s}_t^{t-1} - \ddot{C}a_{t-1}), \\ V_t^{t} & = & V_t^{t-1} - K_t\ddot{A}V_t^{t-1}. \end{array}$$

To compute $\hat{\vec{s}}_t \equiv \vec{s}_t^T$ and $P_t \equiv V_t^T + \vec{s}_t^T(\vec{s}_t^T)^\top$, we perform a set of backward recursions using

$$\begin{array}{lcl} J_{t-1} & = & V_{t-1}^{t-1}D^\top(V_t^{t-1})^{-1}, \\ \hat{\vec{s}}_{t-1}^T & = & \vec{s}_{t-1}^{t-1} + J_{t-1}(\hat{\vec{s}}_t^T - D\hat{\vec{s}}_{t-1}^{t-1} - Ea_{t-1}), \\ V_{t-1}^T & = & V_{t-1}^{t-1} + J_{t-1}(V_t^T - V_t^{t-1})J_{t-1}^\top). \end{array}$$

Furthermore, $P_{t,t-1} \equiv V_{t,t-1}^T + \vec{s}_t^T(\vec{s}_{t-1}^T)^\top$ can be obtained through the backward recursions:

$$V_{t-1,t-2}^T = V_{t-1}^{t-1}J_{t-2}^\top + J_{t-1}(V_{t,t-1}^T - DV_{t-1}^{t-1})J_{t-2}^\top.$$

G More Estimation Details for General Nonlinear Models

The generative model p_θ can be further factorized as follows:

$$\begin{array}{lcl} & & \log p_\theta(\mathbf{y}_{1:T}|\vec{s}_{1:T}, a_{1:T-1}; A, B, C) \\ & = & \log p_\theta(o_{1:T}|\vec{s}_{1:T}; A) + \log p_\theta(r_{1:T}|\vec{s}_{1:T}, a_{1:T-1}; B, C) \\ & = & \sum_{t=1}^T \log p_\theta(o_t|\vec{s}_t; A) + \log p_\theta(r_t|\vec{s}_{t-1}, a_{t-1}; B, C), \end{array} \tag{13}$$

where $p_\theta(o_t|\vec{s}_t; A)$ is modelled by mixture of Gaussians, with A indicating the existence of edges from $\vec{s}_t$ to o_t, and the same for $p_\theta(r_t|\vec{s}_{t-1}, a_{t-1}; B)$, with B indicating the existence of edges from $\vec{s}_{t-1}$ to r_t.

The inference model $q_\phi(\vec{s}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1})$ is factorized as

$$\begin{array}{lcl} & & \log q_\phi(\vec{s}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1}) \\ & = & \log q_\phi(\vec{s}_1|\mathbf{y}_1, a_0) + \sum_{t=1}^T \log q_\phi(\vec{s}_t|\vec{s}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1}), \end{array} \tag{14}$$

where both $q_\phi(\vec{s}_1|\mathbf{y}_1, a_0)$ and $q_\phi(\vec{s}_t|\vec{s}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})$ are modelled with mixture of Gaussians.

The prior $p_0(\vec{s}_{1:T}|a_{1:T-1})$ is factorized as

$$\begin{array}{lcl} & & \log p_0(\vec{s}_{1:T}|a_{1:T-1}) \\ & = & \log p_0(\vec{s}_1) + \sum_{t=2}^T \log p_0(\vec{s}_t|\vec{s}_{t-1}, a_{t-1}), \end{array} \tag{15}$$

where $p_0(\vec{s}_1)$ and $p_0(\vec{s}_t|\vec{s}_{t-1}, a_{t-1})$ are also modelled with mixture of Gaussians.

Thus, the KL divergence can be represented as follows:

$$\begin{array}{lcl} & & \text{KL}\big(q_\phi(\vec{s}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1})\|p_0(\vec{s}_{1:T})\big) \\ & = & \text{KL}\big(q_\phi(\vec{s}_1|\mathbf{y}_1, a_0)\|p_0(\vec{s}_1)\big) + \sum_{t=2}^T \mathbb{E}_{q_\phi}\big[\text{KL}\big(q_\phi(\vec{s}_t|\vec{s}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})\|p_0(\vec{s}_t|\vec{s}_{t-1})\big)\big]. \end{array} \tag{16}$$

The transition dynamics p_γ is factorized as

$$\log p_\gamma(\vec{s}_{1:T}|a_{1:T-1}; D, E) = \sum_{t=1}^T \log p_\gamma(\vec{s}_t|\vec{s}_{t-1}, a_{t-1}; D, E), \tag{17}$$

with $\vec{s}_t|\vec{s}_{t-1}$ modelled with mixture of Gaussians.

Figure 10 gives the diagram of the neural network architecture. We use variational auto-encoders [3] to learn a compact latent-state representation. More specifically, the encoder, which is used to learn the inference model $q_\phi(\vec{s}_t|\vec{s}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})$, includes a Long Short-Term Memory (LSTM [4]) to encode the sequential information with output h_t and a Mixture Density Network (MDN [5]) to output the parameters of the mixture of Gaussians. At each time instance, the input $\langle o_{t+1}, r_{t+1}, a_t\rangle$ is projected to the encoder and a sample of $\vec{s}_{t+1}$ is inferred from q_ϕ as output. The generated sample further acts as an input to the decoder, together with a_{t+1}, and the decoder outputs $\hat{o}_{t+1}$ and $\hat{r}_{t+2}$.

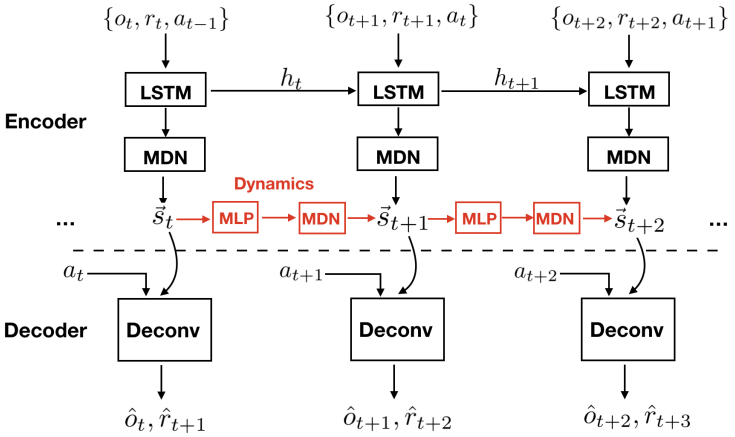

Figure 10: Diagram of neural network architecture to learn state representations. The corresponding structural constraints are involved in “Deconv" and “MLP".

H Algorithm for More Complex Environments

In complex environments, one may update the parameters in Eq. (2) every few iterations. In particular, instead of applying random control signals to record data for model estimation, later iterations may use the learned policy to generate data and further update model parameters. With this way, the agent can better explore the environment and collect new observations, so that the model estimation can be improved and refined over time. The detailed algorithm is given in Algorithm 2.

Algorithm 2 Policy Learning with ASRs in More Complex Environments

Randomly initialize neural networks.
Initialize replay buffer R.
Initialize a random process $\mathcal{N}$ for action exploration.
for iteration = 1,$\cdots$, N **do**
  if iteration == 1 **then**
    Apply random control signals $\mathcal{N}$ and record multiple rollouts.
  else
    Apply the current policy with exploration noise, record multiple rollouts, and attach it to previous collections.
  end if
  Estimate the model given in Eq. (2) with the recorded data (according to Section 4).
  Identify the index of ASRs according to the learned graph structures and the criteria in Proposition 1.
  for episode = 1,$\cdots$,M **do**
    Receive initial observations o_1 and r_1.
    Infer the posterior $q_\phi(\vec{s}_1^{\text{ASR}}|o_1)$ and sample $\vec{s}_1^{\text{ASR}}$.
    for t = 1,$\cdots$, T **do**
      Select action $a_t = \pi(\vec{s}_t) + \mathcal{N}_t$ according to the current policy and exploration noise.
      Execute action a_t and receive reward r_{t+1} and observation o_{t+1}.
      Infer the posterior $q_\phi(\vec{s}_{t+1}^{\text{ASR}}|o_{\leq t+1}, r_{\leq t+1}, a_{\leq t})$ and sample $\vec{s}_{t+1}^{\text{ASR}}$.
      Store transition $(\vec{s}_t^{\text{ASR}}, a_t, r_{t+1}, \vec{s}_{t+1}^{\text{ASR}})$ in R
      Sample a random minibatch of N transitions $(\vec{s}_i^{\text{ASR}}, a_i, r_{i+1}, \vec{s}_{i+1}^{\text{ASR}})$ from R.
      Update network parameters using a specified RL algorithm.
    end for
  end for
end for

I More details about Car Racing Experiment

Car racing (Figure 11) is a continuous control task with three continuous actions: steering left/right, acceleration, and brake. Reward is -0.1 every frame and $+1000/N$ for every track tile visited, where N is the total number of tiles in track. It is obvious that the car racing environment is partially observable, because by just looking at the current frame, we can tell the position of the car, but we know neither its direction nor velocity that are essential for controlling the car.

Ablation Study We further performed ablation studies on latent dynamics prediction; that is, we compared with the case when the transition dynamics (Eq. 17) are not explicitly involved. Figure 13 shows that by explicitly modelling the transition dynamics (denoted by *with LDP*), the cumulative reward has an obvious improvement over the one without modelling the transition dynamics (denoted by *without LDP*).

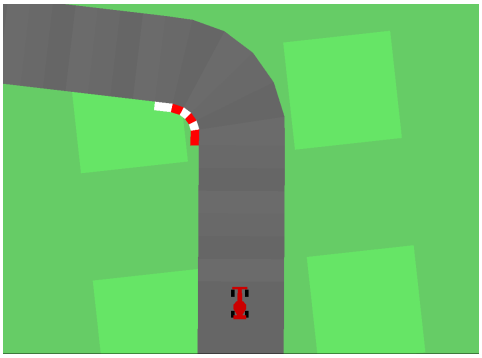

Figure 11: An illustration of Car Racing environment.

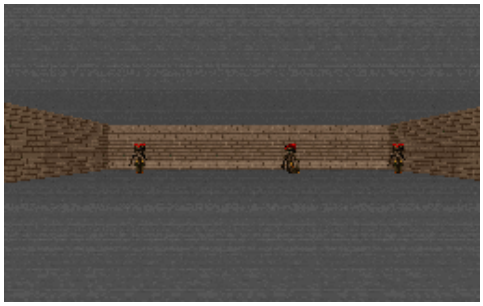

Figure 12: An illustration of VizDoom *take cover* scenario.

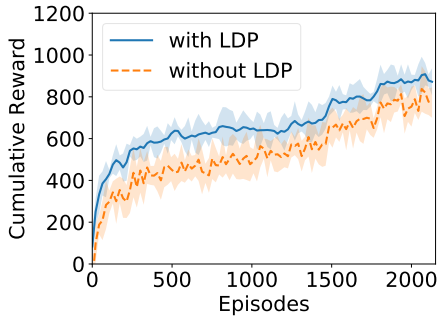

Figure 13: Ablation study of latent dynamics prediction (LDP).

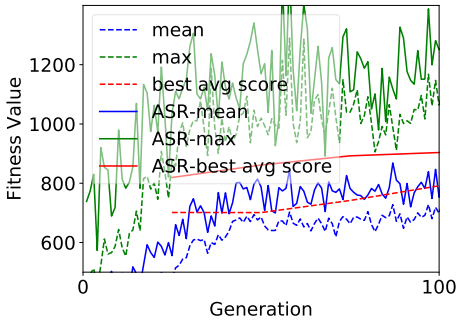

Figure 14: Fitness Value of ASRs applied to world models.

J More Details about VizDoom Experiment

We also conducted the experiments in VizDoom [6]. VizDoom provides a lot of scenarios and we chose the *take cover* scenario (Figure 12) among them which was also used in world models. Different from car racing, *take cover* is a discrete control problem with two actions: move left and move right. Reward is +1 at each time step while alive and the cumulative reward is defined to be the number of time steps the agent manages to stay alive during a episode. Therefore, in order to survive as long as possible, the agent has to learn how to avoid fireballs shot by monsters from the other side of the room. In this task, *solving* is defined as attaining the average survival time of greater than 750 time steps over 100 consecutive episodes, each running for a maximum of 2100 time steps.

In this experiment, following the same setting in [7], we collected a dataset of 10k random rollouts of the environment, each consisting of 500 time steps. The dimensionality of latent state $\vec{s}_t$ is set

to $d_s = 32$. We also set $\lambda_1 = 1$, $\lambda_2 = 6$, $\lambda_3 = 10$, and $\lambda_4 = 0.1$. By tuning thresholds, we finally reported all the results on the 20-dim ASRs (threshold=0.02) which achieved the best results in all the experiments.

Considering that in the *take over* scenario the action space is discrete and ASRs are Markovian, we applied the widely used DQN [8] on ASRs for policy learning. In addition to a natural comparison with the results on SSSs and on original observations alone, we further compared with another common approach to POMDPs: DRQN [9]. As shown in Figure 8 in the main text, DQN on ASRs outperforms DRQN on observations by around 400 on average in terms of cumulative reward.

We also applied ASRs to world models, where CMA-ES is used to optimise the parameters of the controller. As shown in Figure 14, it is evident to see that our method with ASRs (denoted by *ASR-**) achieves a better performance at a faster rate.

K Model Architectures

In the car racing experiment, the encoder consists of three components: a preprocessor, an LSTM, and an MDN. The preprocessor architecture is presented in Figure 15, which takes as input the images, actions and rewards, and its output acts as the input to LSTM. We used 256 hidden units in the LSTM and used a five-component Gaussian mixture in the MDN. The decoder also consists of three components: a current observation reconstructor (Figure 16), a next observation predictor (Figure 17), and a reward predictor (Figure 18). The architecture of the transition/dynamics is shown in Figure 19, and its output is also modelled by an MDN with a five-component Gaussian mixture. In the VizDoom experiment, we used the same architectures except that the LSTM has 512 hidden units and the action has one dimension. It is worth emphasising that we applied weight normalization to all the parameters of the architectures above except for the five matrices A, B, C, D, and E.

In DDPG, both actor network and critic network are modelled by two fully connected layers of size 300 with ReLU and batch normalisation. Similarly, in DQN [8] on both ASRs and SSSs, the Q network is also modelled by two fully connected layers of size 300 with ReLU and batch normalisation. However, in DQN on observations, it is modelled by three convolutional layers (i.e., relu conv $32 \times 8 \times 8 \longrightarrow$ relu conv $64 \times 4 \times 4 \longrightarrow$ relu conv $64 \times 3 \times 3$) followed by two additional fully connected layers of size 64. In DRQN [9] on observations, we used the same architecture as in DQN on observations but padded an extra LSTM layer with 256 hidden units as the final layer.

In each experiment of world models, the controller is parameterized by a fully connected layer whose size is the dimensionality of latent states $\vec{s}_t^{\text{ASR}}$. In this case, it has much fewer parameters than the original controller in the world models.

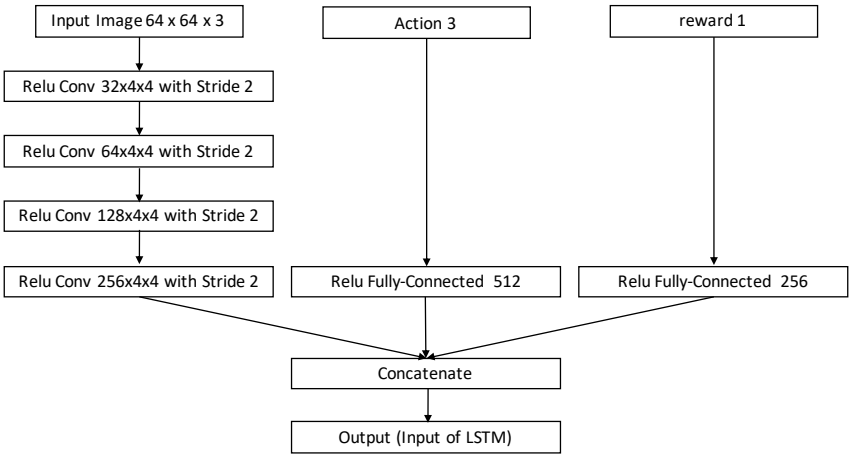

Figure 15: Network architecture of preprocessor.

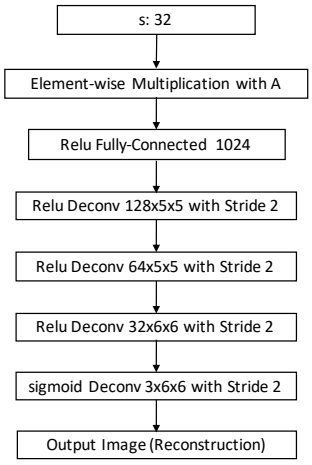

Figure 16: Network architecture of observation reconstruction.

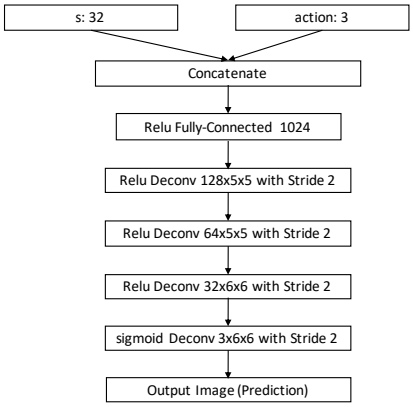

Figure 17: Network architecture of observation prediction.

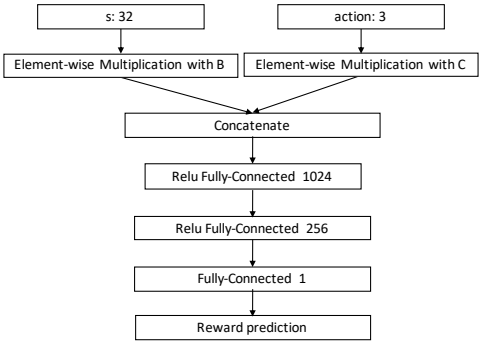

Figure 18: Network architecture of reward.

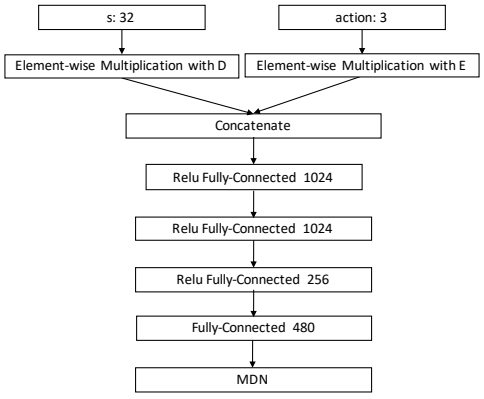

Figure 19: Network architecture of transition/dynamics.

References

[1] K. Zhang and A. Hyvärinen. A general linear non-gaussian state-space model: Identifiability, identification, and applications. In *Asian Conference on Machine Learning*, pages 113–128, 2011.

[2] Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. *Technical Report CRG-TR-96-2, University of Toronto, Dept. of Computer Science*, 1996.

[3] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] C. M. Bishop. Mixture density networks. In *Technical Report NCRG/4288, Aston University, Birmingham, UK*, 1994.

[6] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The best paper award.

[7] D. Ha and J. Schmidhuber. World models. In *Advances in Neural Information Processing Systems*, 2018.

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

[9] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.