Approximate information state for partially observed systems

Jayakumar Subramanian*
Department of Electrical & Computer Engineering
McGill University
Montreal, QC H3A 0E9, Canada
jayakumar.subramanian@mail.mcgill.ca

Aditya Mahajan
Department of Electrical & Computer Engineering
McGill University
Montreal, QC H3A 0E9, Canada
aditya.mahajan@mcgill.ca

Abstract

The standard approach for modeling partially observed systems is to model them as partially observable Markov decision processes (POMDPs) and obtain a dynamic program in terms of a belief state. The belief state formulation works well for planning but is not ideal for online reinforcement learning because the belief state depends on the model and, as such, is not observable when the model is unknown. In this paper, we present an alternative notion of an information state for obtaining a dynamic program in partially observed models. In particular, an information state is a sufficient statistic for the current reward which evolves in a controlled Markov manner. We show that such an information state leads to a dynamic programming decomposition. Then we present a notion of an approximate information state and present an approximate dynamic program based on the approximate information state. Approximate information state is defined in terms of properties that can be estimated using sampled trajectories. Therefore, they provide a constructive method for reinforcement learning in partially observed systems. We present one such construction and show that it performs better than the state of the art for three benchmark models.

1 Introduction

The theory of Markov decision processes focuses primarily on systems with full state observation. When systems with partial state observations are considered, they are converted to systems with full state observations by considering the belief state (which is the posterior belief on the state of the system given the history of observations and actions). Although this leads to an explosion in the size of the state space, the resulting value function has a nice property—it is piecewise linear and convex in the belief state [22]—which is exploited to develop efficient algorithms to compute the optimal policy [15, 21]. Thus, for planning, there is little value in studying alternative characterizations of partially observed models. However, the belief state formulation is not as nice a fit for online reinforcement learning. Part of the difficulty is that the construction of the belief state depends on the system model. So, when the system model is unknown, the belief state cannot be constructed

*A version of this paper has been accepted for publication in IEEE CDC 2019.

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

using the observations. Therefore, critic based methods are not directly applicable. There are some results that circumvent this difficulty [3, 17, 12]. However, many of the recent results suggest that using RNNs (Recurrent Neural Networks [19]) or LSTMs (Long Short Term Memories [14]) for modeling the policy function (actor) and/or the action-value function (critic) works for reinforcement learning in partially observed systems [2, 25, 26, 10, 11, 1]. In this paper, we present a rigorous theory for planning and learning in partially observed models using the notions of information state and approximate information state. We then present numerical experiments that show that the approximate information state based works well on benchmark models.

2 Model

A general system with partial observations may be represented using the following stochastic input-output model. Consider a system that takes two inputs: a control input $U_t \in \mathcal{U}$ and a stochastic input $W_t \in W$ and generates two outputs: an observation $Y_t \in \mathcal{Y}$ and a real-valued reward R_t. The spaces $\mathcal{W}, \mathcal{U}$, and $\mathcal{Y}$ are Banach spaces and the stochastic inputs $(W_1, \dots, W_T)$ are independent random variables defined on a common probability space. For ease of exposition, we ignore measurability and present our main arguments informally. We assume that $\mathcal{W}, \mathcal{U}$, and $\mathcal{Y}$ are finite sets. The arguments can be made rigorous using standard methods [13].

Formally, we assume that there are observation functions $\{f_t\}_{t=1}^T$ and reward functions $\{r_t\}_{t=1}^T$ such that $Y_{t+1} = f_t(Y_{1:t}, U_{1:t}, W_t) \quad \text{and} \quad R_t = r_t(Y_{1:t}, U_{1:t}, W_t)$.

An agent observes the history $H_t = (Y_{1:t}, U_{1:t-1})$ of observations and control inputs until time t and chooses the control input $U_t = \pi_t(H_t)$ according to some history dependent policy $\pi := \{\pi_t\}_{t=1}^T$. The objective of the agent is to choose a policy π to maximize the performance $J(\pi)$:

$$J(\pi) = \mathbb{E}^\pi \left[\sum_{t=1}^T R_t \right]. \tag{1}$$

2.1 A dynamic programming decomposition

In this section, we present a dynamic program for (1) which uses the history of observations and actions as state. Such a dynamic program is not efficient for computing the optimal policy but it will serve as a reference for the rest of the analysis.

First consider the dynamic program for computing the value of any policy π. In particular, define the *reward-to-go* function as

$$J_t(h_t; \pi) := \mathbb{E}^\pi \left[\sum_{s=t}^T R_s \;\middle|\; H_t = h_t \right]. \tag{2}$$

From definitions in (1) and (2), we have $J(\pi) = \mathbb{E}[J_1(H_1; \pi)]$. Thus, the dynamic program (3) gives a recursive method to compute $J(\pi)$. Let $J_{T+1}(h_{T+1}; \pi) := 0$. Then, the reward to go functions can be computed recursively as follows:

$$J_t(h_t; \pi) \overset{(a)}{=} \mathbb{E}^\pi \left[R_t + \mathbb{E}\left[\sum_{s=t+1}^T R_s \;\middle|\; H_{t+1} \right] \;\middle|\; H_t = h_t \right] = \mathbb{E}^\pi \big[R_t + J_{t+1}(H_{t+1}; \pi) \;\big|\; H_t = h_t \big], \tag{3}$$

where (a) follows from the towering property of conditional expectation and the fact that $H_t \subseteq H_{t+1}$. Note that $J_t(h_t; \pi)$ only depends on the future policy $(\pi_t, \dots, \pi_T)$ and not on the past policy $(\pi_1, \dots, \pi_{t-1})$.

Now, recursively define the following *value functions*. $V_{T+1}(h_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$:

$$Q_t(h_t, u_t) = \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, U_t = u_t] \tag{4}$$

$$V_t(h_t) = \max_{u_t \in \mathcal{U}} Q_t(h_t, u_t). \tag{5}$$

Theorem 1 *A policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if it satisfies*

$$\pi_t(h_t) \in \arg\max_{u_t \in \mathcal{U}} Q_t(h_t, u_t). \tag{6}$$

Proof is provided in the Appendix.

2.2 Information state and a simplified dynamic program

Let $\mathcal{F}_t = \sigma(H_t)$ denote the filtration generated by the history of observations and control actions.

Definition 1 An information state $\{Z_t\}_{t\geq 1}$, $Z_t \in \mathcal{Z}$, is an $\mathcal{F}_t$ adapted process (therefore, there exist functions $\{\vartheta_t\}_{t=1}^T$ such that $Z_t = \vartheta_t(H_t)$) that satisfies the following properties:

(P1) **Sufficient for performance evaluation**, i.e.,

$$\mathbb{E}[R_t \mid H_t = h_t, U_t = u_t] = \mathbb{E}[R_t \mid Z_t = \vartheta_t(h_t), U_t = u_t].$$

(P2) **Sufficient to predict itself**, i.e., for any Borel subset A of $\mathcal{Z}$,

$$\mathbb{P}(Z_{t+1} \in A \mid H_t = h_t, U_t = u_t) = \mathbb{P}(Z_{t+1} \in A \mid Z_t = \vartheta_t(h_t), U_t = u_t).$$

There is no restriction on the space $\mathcal{Z}$, although an information state is useful only when the space $\mathcal{Z}$ is "small" in an appropriate sense. We have assumed that the space $\mathcal{Z}$ is time-homogeneous for convenience. In some situations, it may be more convenient to construct an information state which takes values in spaces that are changing with time. For some models, instead of (P2), it is easier to verify the following stronger conditions:

(P2a) **Evolves in a state-like manner**, i.e., there exist measurable functions $\{\varphi_t\}_{t=1}^T$ such that

$$Z_{t+1} = \varphi_t(Z_t, Y_{t+1}, U_t).$$

(P2b) **Is sufficient for predicting future observations**, i.e., for any Borel subset A of $\mathcal{Y}$,

$$\mathbb{P}(Y_{t+1} \in A \mid H_t = h_t, U_t = u_t) = \mathbb{P}(Y_{t+1} \in A \mid Z_t = \vartheta_t(h_t), U_t = u_t).$$

Proposition 1 *(P2a) and (P2b) imply (P2).* □

Proof is provided in the Appendix.

Note that $Z_t = H_t$ is always an information state, so an information state always exists. It is straightforward to show that if we construct a state space model for the above input-output model, then the belief on the state given the history of observations and controls is an information state. We present an example of a non-trivial information state that is much simpler than the belief state:

Example 1 (Machine Maintenance) Consider a machine which can be in one of n ordered states where the first state is the best and the last state is the worst. The production cost increases with the state of the machine. The state evolves in a Markovian manner. At each time, an agent has the option to either run the machine or stop and inspect it for a cost. After inspection, s/he may either repair it (at a cost that depends on the state) or replace it (at a fixed cost). The objective is to identify a maintenance policy to minimize the cost of production, inspection, repair, and replacement.

Let τ denote the time of last inspection and S_τ denote the state of the machine after inspection, repair, or replacement. Then, it can be shown that $(S_\tau, t-\tau)$ is an information state for the system. □

The main feature of an information state is that one can always write a dynamic program based on an information state.

Theorem 2 *Let $\{Z_t\}_{t=1}^T$ be an information state. Recursively define value functions $\{\tilde{V}_t\}_{t=1}^{T+1}$, where $\tilde{V}_t \colon Z_t \mapsto \mathbb{R}$ as follows: $\tilde{V}_{T+1}(z_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\begin{aligned} \tilde{Q}_t(z_t, u_t) &= \mathbb{E}[R_t + \tilde{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, U_t = u_t] \\ \tilde{V}_t(z_t) &= \max_{u_t \in \mathcal{U}} \tilde{Q}_t(z_t, u_t). \end{aligned} \tag{7}$$

Then, we have the following:

$$Q_t(h_t, u_t) = \tilde{Q}_t(\vartheta_t(h_t), u_t) \text{ and } V_t(h_t) = \tilde{V}_t(\vartheta_t(h_t)). \tag{8}$$

Proof is provided in the Appendix.

Remark 1 In light of Theorem 2, an information state may be viewed as a generalization of the traditional notion of state [18, 29]. Traditionally, the state of an input-output system is sufficient for input-output mapping. In contrast, the information state is sufficient for dynamic programming. The notion of information state is also related to sufficient statistics for optimal control [23]. However, in contrast to [23], we do not assume a state space model for the underlying system so it is easier to develop reinforcement learning algorithms using our notion of an information state. □

Coming back to Example 1, Theorem 2 shows that we can write a dynamic program for that model using the information state $(S_\tau, t-\tau)$, which takes values in a countable set. This countable state dynamic program is considerably simpler than the standard belief state dynamic program typically used for that model. Another feature of the information state formulation is that the information state $(S_\tau, t-\tau)$ does not depend on the transition probability of the state of the machine or the cost of inspection or repair. Thus, if these model parameters were unknown, we can use a standard reinforcement learning algorithm to find an optimal policy which maps $(S_\tau, t-\tau)$ to current action.

Given these benefits of a good information state, it is natural to consider a data-driven approach to identify an information state. An information state identified from data will not be exact and it is important to understand what is the loss in performance when using an approximate information state. In Sec. 3, we present a notion of approximate information state and bound the approximation error.

3 Approximate information state (AIS)

Roughly speaking, a compression of the history is an approximate information state if it approximately satisfies (P1) and (P2). This intuition can be made precise as follows.

Definition 2 Given positive numbers ε and δ, an (ε,δ)-approximate information state $\{\hat{Z}_t\}_{t=1}^T$, where $\hat{Z}_t$ takes values in a in a Polish metric space $(\hat{\mathcal{Z}}, d)$, is an $\mathcal{F}_t$ adapted process (therefore, there exist functions $\{\widehat{\vartheta}_t\}_{t=1}^T$ such that $\hat{Z}_t = \widehat{\vartheta}_t(H_t)$) that satisfies the following properties:

(AP1) **Sufficient for approximate performance evaluation**, i.e.,

$$\big|\mathbb{E}[R_t \mid H_t = h_t, U_t = u_t] - \mathbb{E}[R_t \mid \hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t]\big| \le \varepsilon.$$

(AP2) **Sufficient to predict itself approximately**. For any Borel subset A of $\hat{\mathcal{Z}}$ define, $\mu_t(A) = \mathbb{P}(\hat{Z}_{t+1} \in A \mid H_t = h_t, U_t = u_t)$ and $\nu_t(A) = \mathbb{P}(\hat{Z}_{t+1} \in A \mid \hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t)$. Then,

$$\mathcal{K}(\mu_t, \nu_t) \le \delta,$$

where $\mathcal{K}(\cdot,\cdot)$ denotes the Wasserstein or Kantorovich-Rubinstein distance between two distributions. □

Remark 2 Kantorovich-Rubinstein duality [24] states that for any probability measures μ and ν on $\mathcal{X}$,

$$\mathcal{K}(\mu,\nu) = \sup_{\|f\|_{\text{Lip}} \le 1} \Big| \int_{\mathcal{X}} f d\mu - \int_{\mathcal{X}} f d\nu \Big|$$

where $\|f\|_{\text{Lip}}$ denotes the Lipschitz constant of a function f (with respect to the metric d). This along with (P2) imply that for a Lipschitz continuous function $\hat{V}\colon \hat{\mathcal{Z}} \to \mathbb{R}$ with Lipschitz constant L_V (w.r.t. metric d), $\big|\mathbb{E}[\hat{V}(\hat{Z}_{t+1})|H_t = h_t, U_t = u_t] - \mathbb{E}[\hat{V}(\hat{Z}_{t+1})|\hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t]\big| \le L_V\delta$. □

Our main result is that one can write an approximate DP based on an approximate information state.

Theorem 3 *Let $\{\hat{Z}_t\}_{t=1}^T$ be an (ε,δ)-approximate information state. Recursively define value functions $\{\widehat{V}_t\}_{t=1}^{T+1}$, where $\widehat{V}_t\colon \hat{Z}_t \mapsto \mathbb{R}$ as follows: $\widehat{V}_{T+1}(\hat{z}_{T+1}) = 0$ and for $t \in \{T,\dots,1\}$:*

$$\begin{aligned} \widehat{Q}_t(\hat{z}_t, u_t) &= \mathbb{E}[R_t + \widehat{V}_{t+1}(\hat{Z}_{t+1}) \mid \hat{Z}_t = \hat{z}_t, U_t = u_t] \\ \widehat{V}_t(\hat{z}_t) &= \max_{u_t \in \mathcal{U}} \widehat{Q}_t(\hat{z}_t, u_t). \end{aligned} \tag{9}$$

Suppose $\widehat{V}_t$ is Lipschitz continuous with Lipschitz constant L_V. Then, we have the following:

$$\begin{aligned} |Q_t(h_t,u_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), u_t)| &\le (T-t)(\varepsilon + L_V\delta) + \varepsilon \\ |V_t(h_t) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| &\le (T-t)(\varepsilon + L_V\delta) + \varepsilon. \end{aligned} \tag{10}$$

Proof is provided in the Appendix.

Based on Prop. 1, we can provide an alternative characterization of an approximate information state, which is given in the Appendix.

Corollary 1 *Suppose $\{Z_t\}_{t=1}^T$ is an information state and $\{\hat{Z}_t\}_{t=1}^T$ is an (ε, δ)-approximate information state. Then for any realization h_t of H_t, we have the following:*

$$|Q_t(\vartheta_t(h_t), u_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), u_t)| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon$$
$$|V_t(\vartheta_t(h_t)) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon. \tag{11}$$

PROOF The result follows from Theorems 2 and 3. ∎

Use of total variation distance instead of Wasserstein distance and associated bounds are given in the Appendix. Extension of these bounds to the infinite horizon case is straightforward and is provided in the Appendix.

4 Reinforcement learning using approximate information state

4.1 Constructing an approximate information state

The definition of approximate information state suggests two ways to construct an information state from data: either use $\hat{\vartheta}(h_t)$ to determine an approximate information state that satisfies conditions (AP1) and (AP2) or conditions (AP1), (AP2a), and (AP2b). The first approach is more efficient, but the second is easier to understand. So we describe the latter and the former is described in the Appendix.

Note that training a network requires the control inputs $\{U_t\}_{t\ge1}$. In this section, we assume that the control and the observations have been generated according to a pure exploration policy. In the next section, we will consider the case when policy is being learned along with the approximate information state.

4.1.1 Construction based on (AP1), (AP2a) and (AP2b)

We use two function approximators:

- A recurrent neural network (RNN) or its refinements such as LSTM (Long Short-Term Memory) [14] or GRU (Gated Recurrent Unit) [8] with state $C_{t-1} = \hat{Z}_{t-1}$, inputs (Y_t, U_{t-1}) and output $\hat{Z}_t$. We denote this function approximator by ρ.
- A feed forward network with inputs $(\hat{Z}_t, U_t)$ and output $(\tilde{R}_t, \tilde{\nu}_{t+1})$, where $\tilde{R}_t$ is a prediction of the expected reward and $\tilde{\nu}_{t+1}$ is the prediction of ν_{t+1}, the distribution of the next observation Y_{t+1}. We parameterize $\tilde{\nu}_{t+1}$ as multi-variate Gaussain. We denote this function approximator as ψ.

By construction ρ satisfies (AP2a). To minimize the ε in (AP1), we define the loss functions $\mathcal{L}_R = \frac{1}{B}\sum_{t=1}^{B} \texttt{smoothL1}(\tilde{R}_t - R_t)$, where B is the batch size . To minimize the δ in (AP2), we define the loss function $\mathcal{L}_\nu = -\sum_{t=1}^{B-1} \log(\tilde{\nu}_{t+1}(Y_{t+1}))$, which is the negative log likelihood loss for $\tilde{\nu}_t$ and thus approximates the KL-divergence between μ_t and ν_t. We use the KL-divergence as a surrogate for the Wasserstein distance because: (i) Wasserstein distance is computationally expensive to compute; and (ii) KL-divergence upper bounds the total variation (due to Pinsker's inequality), which in turn upper bounds Wasserstein distance for metric spaces with bounded diameter. To train the networks ρ and ψ, we use a weighted combination of these losses to get a single scalar loss:

$$\mathcal{L}_{\rho,\psi} = \lambda \mathcal{L}_R + (1-\lambda)\mathcal{L}_\nu \tag{12}$$

where $\lambda \in [0,1]$ is a hyperparameter.

4.2 Reinforcement learning

In this section, we present an approach to use the approximate information state for reinforcement learning. Let $\pi_\theta : \hat{Z}_t \mapsto \Delta(U_t)$ be a parametrized stochastic policy, where the parameters θ lie in a closed convex set Θ. For example, π_θ could be a feed forward neural network with input $\hat{Z}_t$ and output to be a $|U_t|$ dimensional vector η, which forms the input to a softmax function.

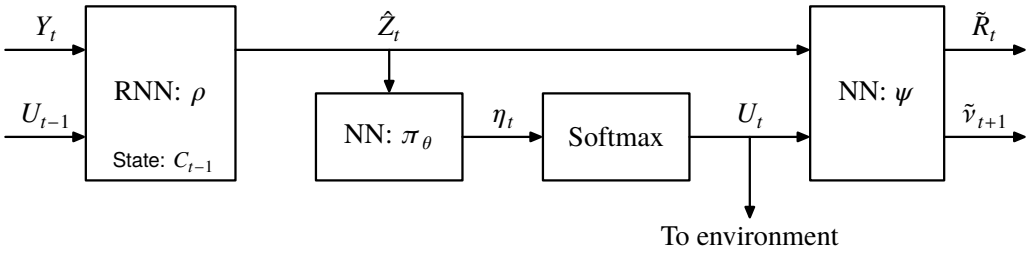
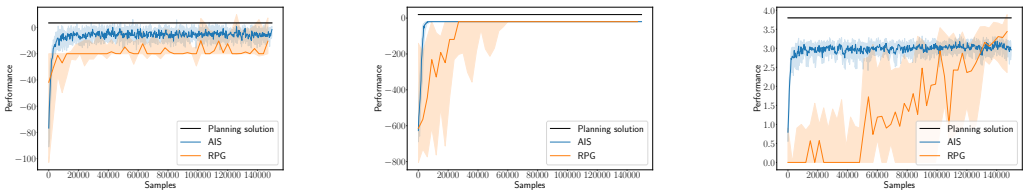


Figure 1: Neural network based function approximators for RL using AIS.



(a) Voicemail problem (b) Tiger problem (c) 4×4 grid problem

Figure 2: Performance versus samples for all examples. The solid line shows the median value and the shaded region shows the region between the first and third quartiles over 25 runs.

An architecture for combining the construction of the approximate information state with reinforcement learning is shown in Fig. 1. In this architecture, we train the networks (ρ, ϕ) and π_θ in parallel using a two time-scale algorithm. In particular, by a slight abuse of notation, let ρ and ψ denote the weights of the corresponding networks. Then,

$$\begin{bmatrix} \rho_{k+1} \\ \psi_{k+1} \end{bmatrix} = \begin{bmatrix} \rho_k \\ \psi_k \end{bmatrix} + a_k \nabla_{\rho,\psi} \mathcal{L}_{\rho,\psi} \text{ and } \theta_{k+1} = \theta_k + b_k \nabla_\theta J(\pi_{\theta_k}),$$

where the learning rates $\{a_k\}_{k\geq 1}$ and $\{b_k\}_{k\geq 1}$ satisfy the standard two time-scale stochastic approximation conditions [5]. We use the approximate information state based reinforcement learning for three small dimensional POMDP benchmarks: voicemail [27], tiger [15] and 4×4 grid [6]. See [20] for the details of the environments. The details of the experiments are provided in the Appendix. The plots for 500 iterations of the algorithm are shown in Fig. 2. We compare our performance with recurrent policy gradient (RPG)[26] algorithm, which is one of the state of the art algorithms for POMDPs. In all three examples, our algorithm performed better than or as good as RPG.

5 Conclusion

In this paper, we present a notion of information state for partially observed systems. We show that an information state is sufficient for dynamic programming. We then relax the definition to describe an approximate information state that can be used to identify an approximately optimal policy. The approximate information state is defined in terms of properties that can be estimated from data, so it can be used to develop sampling based reinforcement learning algorithms. We present one such algorithm and show that it performs better than or comparable to RPG, which is a state of the art reinforcement learning algorithm for POMDPs. The actor only reinforcement learning algorithm presented in this paper is just a proof of concept. It is straight forward to extend standard critic only and actor-critic algorithms using approximate information state by adding a neural network that approximates action-value functions $\hat{Q}^*(\hat{z}, u)$ and $\hat{Q}^\pi(\hat{z}, a)$ in the architecture in Fig. 1. It will be interesting to evaluate how such extensions perform in practice.

Acknowledgments

We are grateful to Raihan Seraj for providing planning and RPG solutions for all three examples in Fig. 2.

References

[1] Andrea Baisero and Christopher Amato. Learning internal state models in partially observable environments;. *Reinforcement Learning under Partial Observability, NeurIPS Workshop*, 2018.

[2] Bram Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2002.

[3] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *JAIR*, 15:319–350, 2001.

[4] D Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Trans. Autom. Control*, 20(3):415–419, 1975.

[5] Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[6] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, 1994.

[7] Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Equivalence relations in fully and partially observable markov decision processes. In *IJCAI*, 2009.

[8] Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[9] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, 2004.

[10] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[11] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv:1512.04455*, 2015.

[12] Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent predictive state policy networks. *arXiv:1803.01489*, 2018.

[13] Onésimo Hernández-Lerma and Jean B Lasserre. *Discrete-time Markov control processes: basic optimality criteria*. Springer, 2012.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[17] Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *NIPS*, 2002.

[18] A. Nerode. Linear automaton transformations. *Proceedings of American Mathematical Society*, 9:541–544, 1958.

[19] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[20] Raihan Seraj. Learning in the presence of partial observability and concept drifts. Master's thesis, McGill University, 2019.

[21] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *AAMAS*, 2013.

[22] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[23] Charlotte Striebel. Sufficient statistics in the optimal control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12:576–592, 1965.

[24] Cédric Villani. *Optimal transport: Old and New*. Springer, 2008.

[25] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007.

[26] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

[27] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[28] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[29] Hans S. Witsenhausen. Some remarks on the concept of state. In Y. C. Ho and S. K. Mitter, editors, *Directions in Large-Scale Systems*, pages 69–75. Plenum, 1976.