Learning a Neural-Network Controller for a Multiplicative Observation Noise System

Vignesh Subramanian
University of California, Berkeley
vignesh.subramanian@berkeley.edu

Moses Won
University of California, Berkeley
moseswon@berkeley.edu

Gireeja Ranade
University of California, Berkeley
ranade@eecs.berkeley.edu

***Abstract*—We consider the stabilization of a linear control system with multiplicative observation noise. Linear strategies have been proven to be ineffective in this system, and non-linear strategies can unboundedly outperform the best linear strategies. The current best-known control strategy is hand-crafted and far from optimal.**

In this paper, we use neural-network-based controllers to narrow the gap between the achievability and the converse for this problem. We propose a periodic controller structure and a greedy training procedure. This enables us to train our controller on a finite horizon problem but learn strategies that outperform the best-known hand-crafted strategy and also generalize, i.e., provide stabilizing control at times beyond the training horizon. Further, we show that for our periodic approach, the learned strategies display a piecewise linear structure and are well approximated by interpretable functions.

***Index Terms*—stochastic control, multiplicative noise, neural networks, stability**

I. Introduction

Simple control systems have been long studied in order to develop an understanding of the fundamental trade-offs involved in communication and control. In 1968 Witsenhausen proposed the Witsenhausen Counterexample [1], where non-linear control strategies could unboundedly outperform linear control strategies [2]. Recent work that used an information-theoretic lens allowed for major insight into the problem and progress towards finding the optimal control strategy (which is still open) [3]–[5]. This progress and insight was preceded by computational studies that examined how various strategies performed on the counterexample; in particular, the strategy of "slopey/soft quantization" that played a role in a provably good strategy in [3] was related to a strategy discovered by Baglietto et al. [6], who used neural networks to find good strategies for the counterexample. Recent work [7] showed that using neural networks for learning control strategies is not a straightforward task and choosing an architecture that favors certain structured strategies is required to escape local minimas that exist due to the non-convex nature of the multi-step control problem.

This paper studies a linear control system:

$$X_{n+1} = aX_n - U_n \tag{1}$$

$$Y_n = Z_n X_n. \tag{2}$$

Here, X_n is the state of the system. The controller may choose the control U_n based on an observation (Y_n) that is corrupted by multiplicative noise ($Z_n \sim \mathcal{N}(0,1)$). This system has been studied before in [8], [9]. The results in [9] provide both an upper and lower bound of the largest a for which this system can be stabilized in a second-moment sense. However, the gap between the bounds is significant, and we believe this is likely due to both of the bounds being loose.

It is notable that the optimal linear strategy for this system is $U_n = 0$ for all n, however non-linear strategies can significantly (and unboundedly) improve on the performance of the linear strategy [9]. A key observation in [9] was that using the controller's memory, i.e. at time n using not only the value of Y_n but also the values of Y_{n-1}, Y_{n-2}, etc. to generate U_n, improved the controller performance.

In this paper, we build on this idea and use neural networks and use the memory of multiple observations to design controllers for this seemingly simple but still-open control problem. We cannot solve this problem using dynamic programming for Markov Decision Processes using the belief state space because we cannot compute the required conditional expectations. We use a periodic control structure and a greedy training procedure coupled with input-output scaling across time. We train our control strategy for a finite-training-horizon, and are able to learn control strategies that generalize to time-horizon well beyond the finite-training-horizon. The learned strategies are well-structured, partly because of our choice of control structure and training procedure. We find we can interpret these strategies in terms of simple features. Finally, we compare strategies across a variety of different parameters and see that while increase in controller memory does benefit the controller performance, there are diminishing returns.

II. Related Work

Our problem formulation builds on many previous ideas in information theory and control that have been discussed in depth in books such as [10]–[12]. Our specific formulation is inspired by the data-rate theorems [13]–[15] as well as the intermittent Kalman Filtering setup [16]; and this formulation was previously discussed in [8], [9]. Additionally, we are inspired by previous works that have studied multiplicative noise including [17]–[21]. Xiao et al. [22] and Xu et al. [23] also consider related problems but effectively restrict their attention to LTI strategies, which are not useful in our problem.

Neural networks have been widely used in the past for system identification as well as to learn good control strate-

gies [24], [25]. There has been significant investigation into the use of modular networks for learning to control dynamical systems [26], [27]. More recently in [28], recurrent neural network based architectures have been used for learning feedback codes in communication system leveraging the noisy feedback from the system. The neural network based feedback codes outperform the best hand-crafted schemes. These works have shown that structured networks can improve training and the overall performance. Our current paper builds on these older results, and our focus is on using neural-network-based strategies to provide new interpretable controls.

III. Problem Formulation

We consider the discrete time system $\mathcal{S}_a$, with initial state $X_0 \sim \mathcal{N}(0,1)$, as given in (1) and (2). At time n, the system state is X_n. A controller observes this state over a multiplicative channel, i.e., $Y_n = Z_n X_n$. We think of Z_n as multiplicative noise. The Z_n's are drawn i.i.d. from a known distribution, however their realizations are unknown to the controller. We focus on $Z_n \sim \mathcal{N}(0,1)$ in this paper, but the ideas generalize. The controller can determine the control at time n, U_n, as a function of the current and past observations, $Y_0, Y_1, \ldots, Y_n$, i.e. $U_n = \pi_n(Y_0, Y_1, \ldots, Y_n)$ where $\pi_n : \mathbb{R}^n \to \mathbb{R}$ is the control strategy at time n. We assume $a \in \mathbb{R}^+$ is fixed and known, and our goal is second-moment stability.

Definition 1. *The system $\mathcal{S}_a$ is stable in second-moment sense if* $\sup_n \mathbb{E}\left[|X_n|^2\right] < \infty$.

We are interested in the question: what is the largest a for which we can stabilize the system in a second-moment sense?

Our goal is to try and close the gap between the achievability and the converse observed in [9]. For simplicity, we focus on the case where $a = 1$, and consider the decay factor of the related system $\mathcal{S}$ given as:

$$X_{n+1} = X_n - U_n, \tag{3}$$
$$Y_n = Z_n X_n. \tag{4}$$

For this system, we define the minimum decay factor as below.

Definition 2. *The minimum decay factor of $\mathcal{S}$ is given as:*

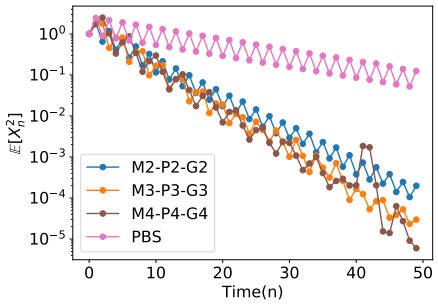
$$d^* = \limsup_{n\to\infty} \inf_{\pi_0,\pi_1,\ldots,\pi_n} \left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}}.$$

Theorem 1 relates the minimum decay factor of system $\mathcal{S}$ to the maximum growth factor that can be stabilized for the system $\mathcal{S}_a$. A proof is provided for completeness in Appendix A.

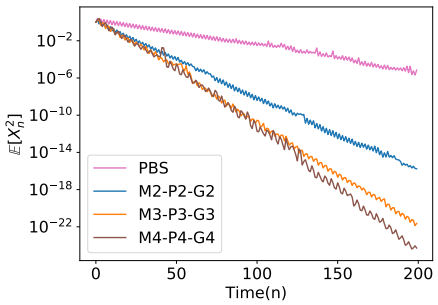
Theorem 1. *Let d^* denote the minimum decay factor for the system $\mathcal{S}$. The system $\mathcal{S}_a$ can be stabilized for all $a < a^*$ and cannot be stabilized for any $a > a^*$ where $a^* = 1/d^*$.*

Note that smaller decay factors correspond to a faster rate of decay of the second moment of the state, X_n, and better performance of the control strategy. We define a few parameters for our control strategy below. First, we say that a strategy uses memory M if it uses the values of observations $Y_n, Y_{n-1}, \ldots, Y_{n-M+1}$ to determine control action U_n. So a memory-1 controller can use only the current observation. Next, we allow the system to periodically cycle through different neural networks as controllers. The parameter P denotes the number of distinct controllers we can use. The value of $n \pmod P$ is used to determine the controller that outputs control action at time n.

We break the training into stages of length G and greedily minimize the second moment of the true state at the end of each stage. More details about the control structure and training procedure are provided in Section VII. In the rest of the sections we use the M-P-G terminology to name our neural network based strategies; as an example M2-P2-G2 refers to a memory-2 control strategy that is 2-periodic and trained in stages of length 2.



(a) $n = 50$ timesteps



(b) $n = 200$ timesteps

Fig. 1. Average second moment vs time for neural-network-based control strategies with memory 2, 3 and 4 and the previous best known strategy (PBS). The memory-2 strategy M2-P2-G2 outperforms the PBS. As we increase M we can achieve faster decay. The gap in the performance for memory-2 and memory-3 strategies is much larger than that between memory-3 and memory-4 strategies.

IV. Main Results

We present our main results in Fig. 1, which shows that neural network based strategies can outperform the previously best known strategy (PBS) [9]. The performance of the PBS plotted here comes from optimizing over the parameters of the strategy from [9].

TABLE I
MAXIMUM GROWTH FACTORS

Strategy	PBS	M1-P2-G4-FIT	M1-P2-G4	M1-P3-G6	M2-P2-G2-FIT	M2-P2-G2	M3-P2-G2	M3-P3-G3	M4-P4-G4
a^*	1.032	1.025	1.026	1.026	1.097	1.097	1.115	1.137	1.156

We tabulate the maximum growth factor that can be stabilized for different strategies in Table I. Restricting to memory-2 control strategies, M2-P2-G2 and can stabilize growth factors up to 1.097 while the PBS from [9] (which also uses memory-2) only stabilizes up to growth factors of 1.032. Increasing the memory of the control strategies further improves the performance of the neural-network-trained controllers and our best strategy can stabilize growth factors up to 1.156. However, there are diminishing returns to increasing memory.

We further show in the following section that these strategies are well-structured.

V. NEURAL-NETWORK-BASED STRATEGIES

This section explores two successful control strategies in detail. We see that these strategies can be understood as linear combination of a few simple features.

A. M1-P2-G4

This control strategy uses only the current observation to compute the control and has a 2-periodic structure. It uses two networks, one for even timesteps and one for odd timesteps, as described in Sec. VII. Since the magnitude of states, observations, and controls for the system decay with time, we need to scale them appropriately as described in detail in Sec. VII. The observations, Y_n, are scaled up to $\widetilde{Y}_n/s_n$ before being fed into the network. The output of the network, $\widetilde{U}_n$, is scaled down to $U_n = s_n\widetilde{U}_n$ before being applied to the system. Fig. 2 shows $\widetilde{U}_n$, the output of the neural network, as a function of $\widetilde{Y}_n$, the input of the neural network, at even and odd time steps.

A first observation about this strategy is that it mostly ignores the sign of $\widetilde{Y}_n$. This makes sense, since the zero-mean multiplicative noise means there is no information in the sign of $\widetilde{Y}_n$. Further, it flips sign at even and odd times and seems to have a "probe" and then "minimize" structure, where at the even timestep it sends a test control that might increase the state magnitude, but uses the observation from this to reduce the magnitude at the following timestep. The system probing suggests there is an element of "active" learning in the strategy.

Based on the shape of the plot, we choose to use a piecewise linear fit (f_{fit}) for the function, and the best fit (using input range $[-5, 5]$) is shown in Fig. 3.

We use the following features for the fit: $\max(-\widetilde{Y}_n+h_L, 0)$, $\max(-\widetilde{Y}_n, 0)$, $\max(\widetilde{Y}_n, 0)$, $\max(\widetilde{Y}_n-h_R, 0)$, and 1 (i.e. a bias term). The parameters h_R and h_L as well as the weights were identified using non-linear least-squares for both the even and odd time control strategies and are listed in Appendix D.

We test the performance of f_{fit} on the actual system and find that it has performance very close to that of the neural network based strategy (see Fig. 4). We see that both strategies can stabilize similar growth factors, 1.025 vs 1.026 (Table I).

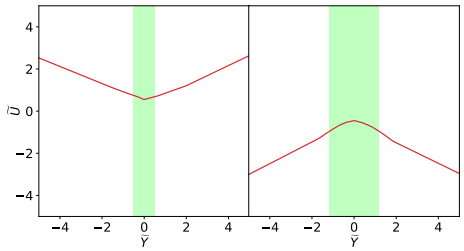


Fig. 2. $\widetilde{U} = f(\widetilde{Y})$ for each neural network. For $n > 30$, 95% of $\widetilde{Y}$'s fed to the neural network lie in the shaded region.

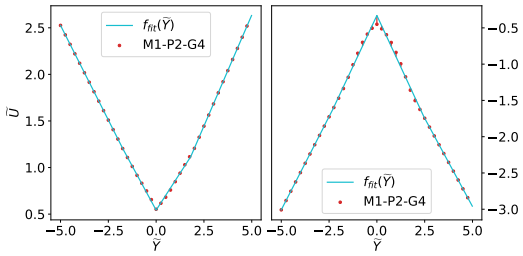


Fig. 3. Piecewise linear fit with four linear segments for the M1-P2-G4 controller. The red dots show generated control $\widetilde{U}_n$ as a function of $\widetilde{Y}_n$. For the first fit, the residuals are bounded in magnitude by 0.025. For the second fit, residual values lie in (-0.125, 0.075).

For both the neural-network-based strategy and the fit strategy the effective function that relates Y_n to U_n changes with time n due to our scaling operation. To check that the scaled inputs $\widetilde{Y}_n$, continue to lie in the range that was used to fit the piecewise linear strategy for different n, we plot the 95^{th} percentile values of $|\widetilde{Y}_n|$ in Fig. 5. We see that in around 30 timesteps the 95^{th} percentile values stabilize to a consistent range depicted by the shaded region in Fig. 2 and the outer regions are used only for times $n < 30$ and for outliers.

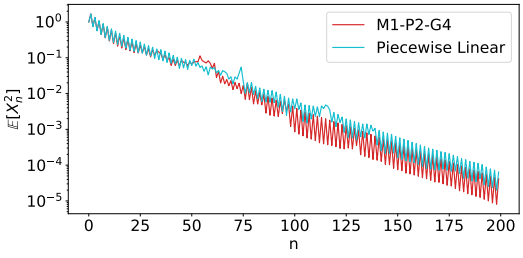


Fig. 4. Performance comparison of the f_{fit} strategy to neural network strategy. Both strategies exhibit similar performance.

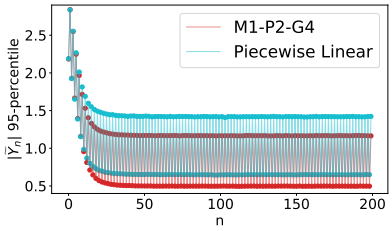


Fig. 5. 95$^{\text{th}}$ percentile value of $|\widetilde{Y}_n|$ (the dots) fed to the neural network with time : There is a difference in the even and odd timesteps as expected for $P = 2$. What is notable is that after about 30 timesteps the inputs stabilize to being in close ranges for both the neural-network-trained controller and the piece-wise linear fit controller. This suggests that good control strategies eventually stabilize the input distribution to the network (up to scaling).

B. M2-P2-G2

Next we consider a memory-2 control strategy with period 2. The successfully trained neural networks use the two most recent scaled observations, $\widetilde{Y}_n$ and $\widetilde{Y}_{n-1}$, to output $\widetilde{U}_n$, and plots for these functions are shown in Fig. 6.

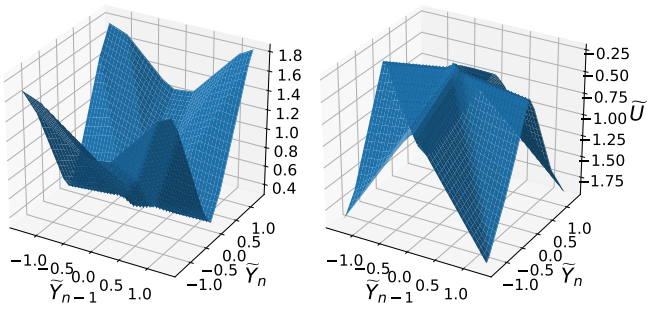


Fig. 6. This figure shows the value of $\widetilde{U}_n$ at odd and even timesteps as a function of $\widetilde{Y}_n$ and $\widetilde{Y}_{n-1}$.

These functions exhibit cross-sectional similarity to the memory-1 strategy in being largely indifferent to the sign of $\widetilde{Y}_n$ and increasing the magnitude of the control based on the magnitude of the observation. However, the relationship between $\widetilde{Y}_n$ and $\widetilde{Y}_{n-1}$ plays an important role in the control strategy now. If we look at the plane spanned by $\widetilde{Y}_n$ and $\widetilde{Y}_{n-1}$ then we can identify four lines in this plane (angles) where the behaviour of the strategy changes. These lines correspond to the creases on the $\widetilde{U}_n$ surface. Motivated by this observation we fit the function (g_{fit}) using the following features: $|\widetilde{Y}_{n-1}|$ and $|\widetilde{Y}_n|$, as well as, $|\cos(\theta_1)\widetilde{Y}_{n-1} - \sin(\theta_1)\widetilde{Y}_n|$, $|\cos(\theta_2)\widetilde{Y}_{n-1} + \sin(\theta_2)\widetilde{Y}_n|$ (i.e. oriented along diagonal lines $Y_n = \pm\cot(\theta)Y_{n-1}$) and 1 (for bias). While we do not yet have a clean explanation for the exact reason why this strategy works, we hope to in future work.

The strategy generated by the fit exhibits similar performance to the network as shown in Fig. 7, and can stabilize similar growth factors as provided in Table I. Like the $M = 1$ strategy, this $M = 2$ strategy exhibits the same zigzag behavior at even and odd timesteps, and the range of scaled inputs also stabilizes after about $n = 30$ timesteps.

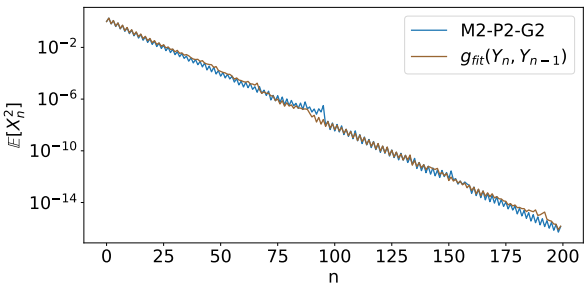


Fig. 7. Performance comparison of the g_{fit} strategy to the neural network generated strategy. The performance of these strategies and the maximum growth factor that these can stabilize are similar.

VI. Effects of the parameters: M, P, G

From section IV it is clear that increasing memory (M), controller period (P) and planning horizon (G) improve performance. This section investigates the effects of these parameters in more detail and Fig. 8 summarizes the results.

We observe that both of the memory-1 strategies, M1-P2-G4 and M1-P3-G6 have similar performance; and increasing the period P or the horizon G while keeping M constant does not seem to improve performance. We notice that when G is a multiple of P, the value of P plays a role in the structure of the strategy. The last action during the period (i.e. when $n = -1 \pmod P$) is always the minimizing control action, whereas earlier actions can be thought of as probing actions. The effect of increasing G to be higher multiples of P is described in Appendix B.

We were unable to train the memory-1 control strategy to work with $P = 1$. Comparing all the strategies M2-P2-G2, M3-P2-G2, M3-P3-G3, we see that the control structure with the most information and degrees of freedom gave the best performance. We believe there are diminishing returns to increasing M, P and G.

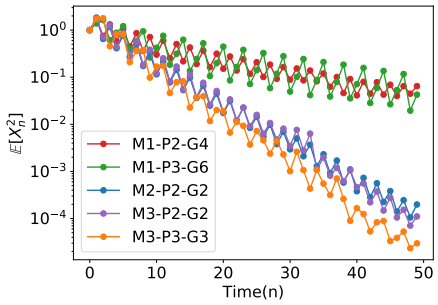
VII. Methods

In this section we list some challenges faced by neural network based control and describe how these motivated the structure of our control strategy and training procedure.

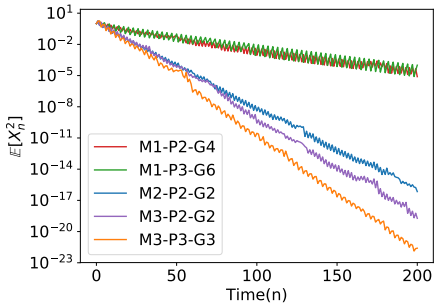
A. Control structure

Each control strategy consists of a set of P networks. At time n the network corresponding to $n \pmod P$ is used to generate the control U_n based on the past M memory of observations $Y_n, \ldots Y_{n-M+1}$. The set of networks over the period P form the control strategy that aims to minimize the second moment of the system state N time-steps in the future. We use a simple one hidden layer architecture for each network with 20 hidden units and ReLU activation.

Since we periodically reuse the same network for control, it is important to consider the inputs and outputs to the network carefully. A good control strategy will decrease the magnitude of the state X_n and thus also the magnitude of the observations Y_n and the required U_n. While an observation



(a) Short rollout up to $n = 50$



(b) Long rollout up to $n = 200$

Fig. 8. Comparison of strategies with different M, P, G values. For memory-1 control strategies $P = 2$ and $P = 3$ result in similar performance but how the strategy alternates between probing and minimization step varies. Increasing M while keeping P, G constant results in better performance but further improvement can be obtained by increasing P, G as well.

value of 0.5 might be typical for timestep 0, it would be a very atypical observation at timestep 100, and hence must be treated differently by the controller. To deal with this we scale inputs and outputs of the networks as described below.

We use an exponential scaling factor s_n given as:

$$s_n = \alpha^{\lfloor \frac{n}{P} \rfloor}.$$

At time n, we scale the observations Y_n by s_n to give $\widetilde{Y}_n$ as:

$$\widetilde{Y}_i = \frac{Y_i}{s_n} \quad i = n - M + 1, \ldots, n.$$

Note that at time n, we scale both the current and previous observations by the scaling factor corresponding to time n in order to preserve the relative order between these observations. Similarly we scale down the network's output $\widetilde{U}_n$ to U_n before applying the control action to the system with $U_n = s_n \widetilde{U}_n$.

We identify the best α by a hyperparameter search and provide the values used in Appendix C. A value of α that is too high or too low leads to poor control strategies that do not generalize. The optimal α is one that leads to a constant second moment of the inputs to the neural networks after the rescaling.

B. Training procedure and network architecture

To train the neural networks we break the training horizon N into stages of length G and greedily minimize the second moment of the true state every G steps. We use truncated backpropagation through time and prevent the flow of gradients across stages [29]. We choose $G = kP$ for some positive integer k. As an example for $M = 2, P = 2, G = 2$ the first stage involves minimizing $\mathbb{E}[X_2^2]$ and the second stage involves minimizing $\mathbb{E}[X_4^2]$. During the second stage we treat Y_2 and Y_1 as fixed constants not dependent on the parameters of the neural networks.

Our control structure and training procedure resembles that of a stateless recurrent neural network with our scaling procedure and periodic control structure performing the role of the state. The structure that we impose makes it easier to train our control strategies as compared to training recurrent neural networks.

We use a training batch size of 4000 and run rollouts of the system (3) for a total training horizon of 24. We train our neural networks for 10000 iterations using the Adam optimizer with a learning rate of 10^{-4}.

C. Testing procedure

To test our controllers we run rollouts of the strategies for batches of 10^6. Note that though this batch size is large it, is not large enough to see certain types of inputs. This might make some strategies appear to be successful even though they should fail. For instance, the probability that all noise realizations are positive for a rollout up to time 50 is roughly 9×10^{-16} and we would require a batch size of around 10^{17} to consistently see such inputs. However, we believe this is not an issue with our strategies since the rate of decrease of the second moment is consistent across time and our batch size is large enough that the empirical test performance for small n is an accurate indicator for the true test performance.

VIII. Conclusions and Future Work

We find that by choosing the appropriate control structure and training procedure we can learn neural network based control strategies that can stabilize a multiplicative observation noise system and outperform hand-crafted strategies. Allowing strategies to use more memory improves performance but has diminishing returns. There is a structure and planning aspect to the learned strategies that can be expressed in terms of simple features. However there are many open questions. Can we exactly quantify how the memory and period of the controllers affects their performance and is there an optimal memory and period to use? Can we better understand the probe and minimize aspect of the controllers? Can this provide insights into the fundamental communication bottlenecks imposed by multiplicative noise in control systems, for example, is there some amount of "active" learning that must be done in these systems by probing? We hope to explore these questions in future work.

Acknowledgments

The authors would like to thank the Siebel Energy Grant for partial funding for this work, and the reviewers for helpful comments that improved the paper.

REFERENCES

[1] H. Witsenhausen, "A counterexample in stochastic optimum control," *SIAM Journal on Control.*, vol. 6, p. 131, 1968.

[2] S. Mitter and A. Sahai, "Information and control: Witsenhausen revisited," *Lecture Notes in Control and Information Sciences*, pp. 281–293, 1998.

[3] P. Grover and A. Sahai, "The vector Witsenhausen problem as assisted interference cancelation," *International Journal on Systems, Control and Communications (IJSCC)*, 2008.

[4] S. Y. Park and A. Sahai, "It may be easier to approximate decentralized infinite-horizon LQG problems," in *Decision and Control (CDC), IEEE Conference on.*, 2012, pp. 2250–2255.

[5] C. Choudhuri and U. Mitra, "On Witsenhausen's counterexample: the asymptotic vector case," *2012 IEEE Information Theory Workshop, ITW 2012*, pp. 162–166, Sep. 2012.

[6] M. Baglietto, T. Parisini, and R. Zoppoli, "Numerical solutions to the Witsenhausen counterexample by approximating networks," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1471–1477, Sep. 2001.

[7] V. Subramanian, L. Brink, N. Jain, K. Vodrahalli, A. Jalan, N. Shinde, and A. Sahai, "Some new numeric results concerning the Witsenhausen counterexample," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct 2018, pp. 413–420.

[8] G. Ranade and A. Sahai, "Non-coherence in estimation and control," in *51st Annual Allerton Conf. on Comm., Control, and Comp.*, 2013.

[9] J. Ding, Y. Peres, G. Ranade, A. Zhai *et al.*, "When multiplicative noise stymies control," *The Annals of Applied Probability*, vol. 29, no. 4, pp. 1963–1992, 2019.

[10] S. Yuksel and T. Basar, *Stochastic Networked Control Systems*. Springer, 2013.

[11] S. Fang, J. Chen, and H. Ishii, *Towards Integrating Control and Information Theories*. Springer, 2016.

[12] A. S. Matveev and A. V. Savkin, *Estimation and Control Over Communication Networks*. Springer Science & Business Media, 2009.

[13] R. W. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 7, pp. 1279–1289, 2000.

[14] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Transactions on Automatic Control*, vol. 49, no. 7, pp. 1056–1068, 2004.

[15] G. N. Nair and R. J. Evans, "Stabilization with data-rate-limited feedback: tightest attainable bounds," *Systems & Control Letters.*, vol. 41, no. 1, pp. 49–56, 2000.

[16] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *Automatic Control, IEEE Transactions on.*, vol. 49, no. 9, pp. 1453–1464, 2004.

[17] M. Athans, R. Ku, and S. Gershwin, "The uncertainty threshold principle: some fundamental limitations of optimal decision making under dynamic uncertainty," *IEEE Transactions on Automatic Control*, vol. 22, no. 3, pp. 491–495, 1977.

[18] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[19] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 163–187, 2007.

[20] N. Elia and J. N. Eisenbeis, "Limitations of linear remote control over packet drop networks," in *Decision and Control (CDC), IEEE Conference on.*, vol. 5. IEEE, 2004, pp. 5152–5157.

[21] G. Ranade and A. Sahai, "Control capacity," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 235–254, 2018.

[22] N. Xiao, L. Xie, and L. Qiu, "Feedback stabilization of discrete-time networked systems over fading channels," *IEEE Transactions on Automatic Control*, vol. 57, no. 9, pp. 2176–2189, 2012.

[23] L. Xu, Y. Mo, L. Xie, and N. Xiao, "Mean square stabilization of linear discrete-time systems over power constrained fading channels," *IEEE Transactions on Automatic Control*, 2017.

[24] A. G. Barto, "Connectionist learning for control: an overview," 1989.

[25] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[26] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton *et al.*, "Adaptive mixtures of local experts." *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[27] R. A. Jacobs and M. I. Jordan, "Learning piecewise control strategies in a modular neural network architecture," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 2, pp. 337–345, 1993.

[28] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, "Deepcode: feedback codes via deep learning," *IEEE Journal on Selected Areas in Information Theory*, vol. PP, pp. 1–1, Apr. 2020.

[29] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, vol. 2, no. 4, pp. 490–501, Dec 1990.

APPENDIX

A. Proof of Theorem 1

For the sake of this proof, let us label the states, observations, multiplicative noise and control actions for the system $\mathcal{S}_a$ as X_n^a, Y_n^a, Z_n^a and U_n^a respectively. We will use the notation X_n, Y_n, Z_n and U_n for the system $\mathcal{S}$. Consider a coupling of the two systems such that $X_0 = X_0^a$ and $Z_n^a = Z_n$ for all n.

Suppose the control action at time n, U_n, for system $\mathcal{S}$ is given by $U_n = \pi_n(Y_0, Y_1, \ldots, Y_n)$. Note that here $\pi_n : \mathbb{R}^n \to \mathbb{R}$ is a deterministic function given the realizations of the observations. Then, we construct the following control actions for system $\mathcal{S}_a$,

$$U_n^a = \pi_n^a(Y_0^a, Y_1^a, \ldots, Y_n^a) \tag{5}$$

$$= a^{n+1}\pi_n\left(Y_0^a, \frac{Y_1^a}{a}, \ldots, \frac{Y_n^a}{a^n}\right). \tag{6}$$

We will prove by induction that under these sets of controls, the true state and observations for the two systems are related as,

$$X_n^a = a^n X_n, \quad Y_n^a = a^n Y_n.$$

Note that by our assumption on the initial state and noise realizations the base case for $n = 0$ is true. Now assume that the claim is true for $n \le k$. We have,

$$\begin{aligned} X_{k+1}^a &= aX_k^a - U_k^a \\ &= a^{k+1}X_k - a^{k+1}\pi_k\left(Y_0^a, \frac{Y_1^a}{a}, \ldots, \frac{Y_k^a}{a^k}\right) \\ &= a^{k+1}X_k - a^{k+1}\pi_k\left(Y_0, Y_1, \ldots, Y_k\right) \\ &= a^{k+1}(X_k - U_k) \\ &= a^{k+1}X_{k+1}. \end{aligned}$$

Further since the noise realizations are same we have

$$Y_{k+1}^a = Z_{k+1}^a X_{k+1}^a = Z_{k+1}a^{k+1}X_{k+1} = a^{k+1}Y_{k+1}.$$

Thus the claim is true for $n = k + 1$ and this completes the inductive proof.

Next we will show that if the minimum decay factor for system $\mathcal{S}_a$ is d^* then system $\mathcal{S}_a$ can be stabilized for $a < 1/d^*$.

Suppose $\pi_0^*, \ldots, \pi_n^*$ minimize $\left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}}$ for each n. Let π_n^* denote the minimizing control action for system $\mathcal{S}$ and consider $(\pi_n^a)^*$ as the control action for system $\mathcal{S}_a$, where $(\pi_n^a)^*$ and π_n^* are related as in Equation (6).

From Definition 2 we have,

$$d^* = \limsup_{n\to\infty}\left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}}.$$

Then by definition of $\limsup$ we have for every $\epsilon > 0$, there exists N such that for all $n \ge N$,

$$d^* \ge \left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}} - \epsilon.$$

Thus,

$$\begin{aligned} (d^* + \epsilon)^{2n} &\ge \left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right) \\ &= \frac{1}{a^{2n}}\mathbb{E}[|X_n^a|^2], \end{aligned}$$

since $\mathbb{E}[|X_0|^2] = 1$. Any $a < 1/d^*$ can be written as $a = (1-\delta)(1/d^*)$ for some $\delta > 0$. Thus we have,

$$\begin{aligned} \mathbb{E}[|X_n^a|^2] &\le a^{-2n}(d^* + \epsilon)^{2n} \\ &= \left(1 - \delta + \frac{\epsilon(1-\delta)}{d^*}\right)^{2n}. \end{aligned}$$

Since this bound holds for any $\epsilon > 0$, taking $\epsilon = \frac{\delta d^*}{2(1-\delta)}$, there exists N such that for all $n \ge N$,

$$\mathbb{E}[|X_n^a|^2] \le \left(1 - \frac{\delta}{2}\right)^{2n} < 1.$$

Further, because $\sup_{n<N}\mathbb{E}[|X_n^a|^2]$ is finite, we conclude that system $\mathcal{S}_a$ is stabilizable.

Next we will show that the system $\mathcal{S}_a$ cannot be stabilized for any $a > \frac{1}{d^*}$. Consider such an $a = \frac{1}{d^*}(1+\gamma)$ for some $\gamma > 0$. Suppose for contradiction there exists a set of control actions $\widetilde{\pi_0^a}, \widetilde{\pi_1^a}, \ldots, \widetilde{\pi_n^a}$ such that $\sup_n \mathbb{E}[|X_n^a|^2]$ is finite. Thus there exists $K < \infty$ such that for all n,

$$\mathbb{E}[|X_n^a|^2] \le K.$$

Further using the set of control actions $\widetilde{\pi_0}, \widetilde{\pi_1}, \ldots, \widetilde{\pi_n}$ where $\widetilde{\pi_n^a}$ and $\widetilde{\pi_n}$ are related as in Equation (6) we have,

$$\begin{aligned} &\mathbb{E}[|X_n^a|^2] \le K \\ \implies\ & a^{2n}\mathbb{E}[|X_n|^2] \le K \\ \implies\ & a^{2n}\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]} \le K \\ \implies\ & \left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}} \le (K)^{\frac{1}{2n}}a^{-1} = (K)^{\frac{1}{2n}}\frac{d^*}{1+\gamma}. \end{aligned} \tag{7}$$

Choose N such that for all $n \ge N$,

$$(K)^{\frac{1}{2n}} < (1+\gamma).$$

Note that such an N exists because K is finite and $\lim_{n\to\infty}(K)^{\frac{1}{2n}} = 1$. Since the upper bound in Equation (7) holds for all n, we have for $n \ge N$,

$$\left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}} < d^*,$$

Thus, we have a set of control actions $\widetilde{\pi_0}, \widetilde{\pi_1}, \ldots, \widetilde{\pi_n}$ such that,

$$\limsup_{n\to\infty}\left(\frac{\mathbb{E}[|X_n|^2]}{\mathbb{E}[|X_0|^2]}\right)^{\frac{1}{2n}} < d^*$$

which is a contradiction since we assumed that d^* was the minimum decay factor. This completes the proof.

TABLE II
α VALUES

Strategy	M1-P2-G4-FIT	M1-P2-G4	M1-P3-G6	M2-P2-G2-FIT	M2-P2-G2	M3-P2-G2	M3-P3-G3	M4-P4-G4
α	0.955	0.955	0.933	0.832	0.832	0.808	0.685	0.551

B. Effect of the parameter G

In Sec. VI we discussed the motivation for setting G as multiples of P. For memory-1 strategies we observed that setting $G = P$ results in overfitting while training leading to poor test performance. Keeping M and P constant and increasing the value of G does not lead to better performance during training but alleviates the problem of overfitting by taking account of the fact that the same networks are being used across multiple periods, while minimizing the cost function. Thus the strategy $M1 - P2 - G4$ performed much better that $M1 - P2 - G2$ during testing.

C. Values for scaling hyperparameter α

We list the alpha values for different strategies in Table. II. We see that strategies that perform better and lead to faster decay of rates correspond to smaller values α. For the fit strategies we use the same values as the original neural network based strategy.

D. Fit strategy for M1-P2-G4

We can express the function that relates the scaled inputs $\widetilde{Y}_n$ to the network to the output of the network $\widetilde{U}_n$ by f_{fit} where,

$$f_{\text{fit}}(\tilde{Y}) = (m_1 - m_2)\max(-\tilde{Y} + h_L, 0) + m_2 \max(-\tilde{Y}, 0) + m_3 \max(\tilde{Y}, 0) + (m_4 - m_3)\max(\tilde{Y} - h_R) + b.$$

The resultant values for the parameters $h_L, h_R, m_1, m_2, m_3, m_4,$ and b for even and odd timestep networks are as follows.

Fit Parameter	Even time	Odd Time
h_L	1.807	2.367
h_R	1.808	2.429
m_1	0.406	-0.512
m_2	0.378	-0.561
m_3	0.317	-0.568
m_4	0.476	-0.486
b	0.542	-0.329

E. Fit strategy for M2-P2-G2

Since this is a memory-2 strategy, we can express the function that relates the scaled inputs $\widetilde{Y}_n, \widetilde{Y}_{n-1}$ to the output $\widetilde{U}_n$ by g_{fit} where,

$$g_{\text{fit}}\left(\widetilde{Y}_n, \widetilde{Y}_{n-1}\right) = k_1 \left|\widetilde{Y}_{n-1}\right| + k_2 \left|\widetilde{Y}_n\right| + k_3 \left|\cos(\theta_1)\widetilde{Y}_{n-1} - \sin(\theta_1)\widetilde{Y}_n\right| + k_4 \left|\cos(\theta_2)\widetilde{Y}_{n-1} + \sin(\theta_2)\widetilde{Y}_n\right| + b.$$

The resultant values for the parameters $k_1, k_2, k_3, k_4, \theta_1, \theta_2,$ and b for even and odd timestep networks are given below.

Fit Parameter	Even time	Odd Time
k_1	-0.455	0.863
k_2	-0.924	1.302
k_3	0.271	-0.548
k_4	0.279	-0.585
θ_1	0.654	0.856
θ_2	0.682	0.870
b	-0.675	0.236