Linear Dynamical Systems as a Core Computational Primitive

Shiva Kaul [1]

Abstract

Single-input, single-output linear dynamical systems (SISO LDS) map a sequence of input numbers to a sequence of output numbers. We present two results which support their use as a building block for more complex RNNs. The first result concerns computational efficiency. We show that reachable SISO LDS, and their gradients, can be computed in parallel across time, so they can be used on very long time series. This is possible because the eigenvectors of the LDS transition matrix have closed-form expressions in terms of the eigenvalues. The second result concerns expressive power. We show a sum of reachable SISO LDS can approximate any reachable, multiple-input, single-output (MISO) LDS, whose inputs are vectors. This construction involves randomly projecting the vectors to a single dimension.

1. Introduction

Linear Dynamical Systems (LDS) are classical RNNs in which the next state $s_{t+1} = As_t + Bx_t$ is a linear function of the current state s_t and input x_t. They are a mainstay of control theory and many engineering applications because their behavior can be easily regulated (Zhou et al., 1996). Under certain conditions, one can guide the system to any internal state by using the appropriate inputs (*reachability*), or may infer the system's state from its input-output behavior (*observability*). Recently, LDS have enjoyed a renaissance in machine learning theory. They are simple testbeds which elucidate gradient descent (Hardt et al., 2016) and the importance of depth (Hardt & Ma, 2016). They capture the behavior of optimization algorithms (Lessard et al., 2016) and establish baseline performance for reinforcement learning (Recht) and online learning (Hazan et al., 2017). Because they are well understood, it would be prudent if LDS were used as a building block for more complex RNNs.

[1]Computer Science Department, Carnegie Mellon University, Pittsburgh PA, USA. Correspondence to: Shiva Kaul <skkaul@cs.cmu.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

However, RNNs (including reachable LDS) suffer from a major computational bottleneck when used on long sequences of data: because the current state of the RNN depends on the previous one, running the RNN is a sequential operation which cannot exploit parallel hardware. This bottleneck has recently motivated many practitioners to abandon RNNs altogether and to model time series by other means. These include hierarchies of (dilated) convolutions (Oord et al., 2016; Gehring et al., 2017), convolutions applied to sliding windows (Miller & Hardt, 2019), and attention mechanisms which encode positions of interest in the input (Vaswani et al., 2017). In these models, highly-parallel convolutions are the key underlying primitive.

1.1. Our Contributions

We find a parametrization of reachable, SISO LDS which is very computationally appealing. The total number of parameters is minimal. No constraints need to be enforced upon the parameters to ensure reachability. Most importantly, forward and backward passes of the LDS may be computed in parallel.

Proposition 1. *Reachable, SISO, n-state LDS can be parametrized by n eigenvalues and a row vector $C \in \mathbb{R}^{1\times n}$, without (nontrivial) constraints. Given the parameters and a length-T sequence of inputs, it is possible to compute the LDS outputs, and their gradients with respect to the parameters, in $O(n(T/p + \log p))$ time on p parallel processors.*

The key to this parametrization is just a bit of linear algebra. In an LDS, all the state variables interact with one another through the transition matrix A. These interactions disappear when A is diagonalized, i.e. when the LDS is run in the basis of its eigenvectors. In this *modal* form, the LDS may be run with a parallel linear recurrence (PLR) solver. However, in that form, it is not possible to enforce reachability. We show that, for reachable SISO LDS, the eigenvectors of the LDS have closed-form expressions in terms of the eigenvalues. By using these expressions, we enforce reachability in the parallelism-friendly modal form.

The previous result applies only to SISO LDS (and probably cannot be generalized to MISO LDS), whereas most input sequences in machine learning are high-dimensional. The next result shows that SISO LDS can be composed to handle higher-dimensional data. Any MISO LDS can be

approximated by the average of k SISO LDS produced by random projections.

Proposition 2. *Let $x_1, \ldots, x_T$ be any sequence of d-dimensional inputs, and let $y_1, \ldots, y_T$ be the corresponding outputs of a reachable MISO LDS with parameters (A, B, C, D). For each $i \in [k]$, let $r_{(i)}$ be a d-dimensional standard normal vector, $x_t^{(i)} = r_{(i)}^T x_t$ be a projected sequence of scalar inputs, and $(A, Br_{(i)}, C, D)$ be the parameters of a SISO LDS producing outputs $y_t^{(i)}$. Let $\hat{y}_t = \frac{1}{k}\sum_{i=1}^{k} y_t^{(i)}$ be the average output. For each $t \leq T$, $\mathbf{E}(y_t - \hat{y}_t)^2 = 2\,||Z_t||_F^2 / k$, where Z_t is defined in (7). Furthermore, the SISO LDS are reachable almost surely.*

$||Z_t||_F^2$ is typically independent of t when A has spectral radius less than 1 (c.f. equation 8).

By making reachable SISO LDS faster to run, and demonstrating their compositional power, we highlight their potential as core primitives in more complex RNNs.

2. Linear Dynamical Systems

Let the input at time $t \in [T]$ be $x_t \in \mathbb{R}^d$. Single-input and multiple-input LDS correspond to $d = 1$ and $d > 1$. An LDS with state size n takes the following form:

$$s_{t+1} = As_t + Bx_t \qquad y_t = Cs_t + Dx_t \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times d}$, $C \in \mathbb{R}^{1\times n}$, $D \in \mathbb{R}^{1\times d}$, $s_t \in \mathbb{R}^{n\times 1}$ and $y_t \in \mathbb{R}$. The internal state s_t changes over the course of time. Larger n make for a more powerful model that is more expensive to run. By recursively unrolling (1), we obtain the following expression for the outputs as a convolution of the inputs:

$$y_t = CA^t s_0 + \sum_{\tau=1}^{t-1} CA^\tau B x_{t-\tau} + Dx_t \tag{2}$$

For notational simplicity, we may omit the term Dx_t, in which case the LDS is called *strictly causal*.

2.1. Reachability and Observability

More significantly, we focus on LDS that are *reachable*, which means that we can make the system do anything we want by supplying the right input.

Definition 1 (Reachability)**.** *A state $s \in \mathbb{R}^n$ is reachable if there is a sequence of inputs $x_1, \ldots, x_T$ which leads to $s_T = s$. An LDS is reachable if every state $s \in \mathbb{R}^n$ is reachable.* [1]

The following characterization of reachability will be useful.

[1] In continuous time, reachability and controllability are equivalent. In discrete time, they are equivalent only when A is nonsingular.

Lemma 1 (Hautus)**.** *An LDS is reachable iff A is nonsingular and, for all $\lambda \in \mathbb{C}$, the $n \times (n + d)$ matrix $[\lambda I - A; B]$ has full rank n.*

A reachable, SISO LDS can be written in the following canonical form:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & \ldots & -a_{n-1} \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{3}$$

An obvious advantage of this form is that the number of parameters reduces from $n^2 + 2n$ to just $2n$: n for the last row of A, and n for the vector C. Another advantage is the state update rule becomes very simple:

$$s_{t+1} = \begin{pmatrix} s_{t,2} \\ \vdots \\ s_{t,n} \\ x_t - \sum_{1\leq i\leq n} a_{i-1} s_{t,i} \end{pmatrix} \tag{4}$$

A related notion to reachability is observability, which means we can determine what is going on inside the system just by observing its input-output behavior.

Definition 2 (Observability)**.** *An LDS is observable if, from any input-output sequence $(x_1, y_1), \ldots, (x_n, y_n)$, the (nonzero) initial state s_0 may be determined.*

Reachability and observability are mathematically dual: the LDS (A, B, C, D) with inputs x and outputs y is reachable iff the LDS (A^T, C^T, B^T, D^T) with inputs y and outputs x is observable. Though our results only apply to reachable systems, it should be possible to develop analogues for observable systems.

2.2. Modal Representation

A can be decomposed in terms of its eigenvalues and eigenvectors: $A = U\Lambda U^{-1}$ where Λ is a diagonal matrix with the eigenvalues and U is a matrix whose columns are corresponding eigenvectors. Another way of writing this is $\Lambda = U^{-1}AU$. Since A is real, it may have complex eigenvalues, but these come in conjugate pairs; if $\lambda_j = a_j + b_j i$ is an eigenvalue, then so too is $\lambda_{j+1} = \overline{\lambda}_j = a_j - b_j i$. Let us rewrite the LDS in the basis of the eigenvectors. Define the state s_t as a rotation of an eigenstate: $s_t = Us'_t$. The LDS is:

$$Us'_{t+1} = AUs'_t + Bx_t \qquad y_t = CUs'_t \tag{5}$$

Defining $B' = U^{-1}B$ and $C' = CU$, we obtain the modal representation of the LDS:

$$\begin{aligned} s'_{t+1} &= U^{-1}AUs'_t + U^{-1}Bx_t = \Lambda s'_t + B'x_t \\ y_t &= C's'_t \end{aligned} \tag{6}$$

3. Approximating MISO with SISO

We present the approximation result first because it is slightly simpler. Let us take $D = 0$ and $s_0 = 0$ for notational simplicity. From the convolution representation (2) and the random construction of the SISO LDS, we find that the approximation is unbiased:

$$\begin{aligned}\mathbf{E}\,\hat{y}_t &= \mathbf{E}\,\frac{1}{k}\sum_i \sum_{\tau=1}^{t-1} CA^{t-1-\tau}Br_{(i)}r_{(i)}^T x_\tau \\ &= \sum_{\tau=1}^{t-1} CA^\tau Br_{(i)}\left(\frac{1}{k}\mathbf{E}r_{(i)}r_{(i)}^T\right)x_{t-\tau} = y_t\end{aligned}$$

Therefore the mean squared error is just the variance:

$$\mathbf{E}\,(y_t - \hat{y}_t)^2 = \mathbf{E}\,(\mathbf{E}\,\hat{y}_t - \hat{y}_t)^2 = \mathbf{V}(\hat{y}_t)$$

By the independence of the $r_{(i)}$, and the cyclic property and linearity of trace, we reduce to the variance of a quadratic in normal variables:

$$\begin{aligned}\mathbf{V}(\hat{y}_t) =& \mathbf{V}\left(\sum_{\tau=1}^{t-1}\mathrm{tr}(CA^\tau B\left(\frac{1}{k}\sum_{i=1}^{k} r_{(i)}r_{(i)}^T\right)x_{t-\tau})\right) \\ =& \frac{1}{k^2}\sum_{i=1}^{k}\mathbf{V}\left(\sum_{\tau=1}^{t-1}\mathrm{tr}(r_{(i)}^T x_{t-\tau}CA^\tau Br_{(i)})\right) \\ =& \frac{1}{k^2}\sum_{i=1}^{k}\mathbf{V}\Bigg(r_{(i)}^T\underbrace{\sum_{\tau=1}^{t-1}x_{t-\tau}CA^\tau B}_{Z_t}r_{(i)}\Bigg)\end{aligned} \tag{7}$$

The inner quadratic is not changed by replacing Z_t, which is asymmetric, with $\bar{Z}_t = \frac{1}{2}(Z_t + Z_t^T)$, which is symmetric, diagonalizable, and shares the same eigenvalues $\lambda_1, \ldots, \lambda_d$. $r_{(i)}$ retains its distribution under the rotation U that diagonalizes $\bar{Z}_t$. We find the variance is just the squared Frobenius norm of Z_t:

$$\begin{aligned}\mathbf{V}\left(r_{(i)}^T\bar{Z}_t r_{(i)}^T\right) =& \mathbf{V}\left(r_{(i)}^T U^T\mathrm{diag}(\lambda)Ur_{(i)}\right) \\ =& \mathbf{V}\left(\sum_{j=1}^{d} r_{(i),j}^2\lambda_j\right) = 2\sum_{j=1}^{d}\lambda_j^2 = 2\,||Z_t||_F^2\end{aligned}$$

To conclude the proof of Proposition 2, we verify that the SISO LDS are almost surely reachable, assuming the MISO LDS is reachable. By Lemma 1, we must show that if $[\lambda I - A; B]$ has full rank for all $\lambda \in C$, then $[\lambda I - A; Br_{(i)}]$ also does, almost surely. This holds because $r_{(i)}$ has density with respect to Lebesgue measure.

In the above proof, we used only equalities, and obtained a result in terms of the somewhat opaque spectrum of Z_t. Here is some intuition for typical magnitude of $||Z_t||_F^2 = \mathrm{tr}(Z_t^T Z_t)$. Suppose each x_t has standard $N(0,1)$ components, as is typical in dynamical systems literature. Also assume that $||A||_2 = \rho < 1$ (i.e. the LDS is *externally stable*), $||B||_2 \leq 1$, and $||C|| \leq 1$. By the definition of the Frobenius norm and independence of each input:

$$\begin{aligned}\mathbf{E}\,\mathrm{tr}(Z_t^T Z_t) &= \mathrm{tr}\sum_{\tau=1}^{t-1} B^T A^{\tau T} C^T\left(\mathbf{E}\,x_{t-\tau}^T x_{t-\tau}\right)CA^\tau B \\ &\leq d\sum_{\tau=1}^{t-1}\rho^{2\tau} \leq d\frac{\rho^2}{1-\rho^2}\end{aligned} \tag{8}$$

4. Parallelizing Reachable SISO LDS

We briefly present a parallelization technique and discuss the difficulties in naively applying it. We then present a linear algebra fact about reachable SISO LDS, which allows the technique to be applied.

4.1. Parallel Linear Recurrences

The following parallel linear recurrence (PLR) algorithm underlies our approach.

Proposition 3. *Let $a_1, \ldots, a_T$ and $z_1, \ldots, z_T$ be (possibly parametrized) sequences of d-dimensional vectors. Let $\odot$ denote entrywise product between two vectors. For $t \in [T]$, the recurrence $h_{t+1} = a_t \odot h_t + z_t$, and its gradient with respect to the parameters, can be computed in $O\left(n\left(\frac{T}{p} + \log p\right)\right)$ depth, which is less than $O(n \cdot T)$ when $p \geq 3$ and $T \gg p$. (Martin & Cundy, 2018)*

We do not have space to describe the details of the algorithm, but we offer some background. The goal of parallelizing a computation is to reduce its depth (i.e. wall clock time) without substantially increasing its overall work (i.e. total processor time.) Parallel computation of $h_1, h_2, \ldots, h_T$ seems difficult when h_{t+1} depends on h_t. Interestingly, parallelization is possible when h_t is a linear recurrence, i.e. when $h_{t+1} = A_t h_t + z_t$ for general $n \times n$ matrices A_t (Blelloch, 1990). Unfortunately, the parallel algorithm involves matrix multiplication, whose $O(n^3)$ work is prohibitive. Martin & Cundy (2018) restrict attention to diagonal matrices A_t, whose multiplication is just $O(n)$ work. They offer a GPU implementation of PLR and devise parallel variants of LSTMs.

4.2. The Dilemma

Note that a modal form LDS (6) is compatible with PLR, and may thereby be parallelized. However, the set of LDS with a modal form is strictly larger than the set of reachable LDS. In many applications, it may be desirable to consider only reachable LDS. (For example, if one is learning an LDS model of robot behavior, there should always be a series of commands that cause it to halt, i.e. reach the origin.)

To ensure reachability, we begin with an LDS in canonical

form (3), whose parameters are the last row of A and the vector C. To attempt to make it compatible with PLR, we diagonalize it with a rotation U, whose columns are the eigenvectors of A, as in (5) and (6). However, this introduces new parameters for the eigenvectors constituting U, which belongs to the set of orthogonal matrices $\mathrm{SO}(n)$. Parametrized in the usual way with n^2 entries, such matrices form a nonconvex set. The Givens reparametrization admits a coordinate-descent algorithm (Shalit & Chechik, 2014), but does not reduce the number of variables. This difficulty is also encountered by (Huang et al., 2017), who develop a different representation for MIMO LDS.

The dilemma is: how do we reparametrize the set of reachable SISO LDS without increasing the total number of parameters, or involving difficult constraints? Following the rotation, A becomes a diagonal matrix Λ whose entries are its eigenvalues, so we could use those as parameters. But what about the eigenvectors U involved in B' and C'?

4.3. Explicit Eigendecomposition

For reachable SISO LDS, the eigenvectors of A can be written in terms of the eigenvalues of A. We don't need any additional parameters for the eigenvectors.

Lemma 2. *Let $\lambda_j \in \mathbb{C}$ be the j'th eigenvalue of A. Its (unnormalized) eigenvector is $u_j = \left[\frac{1}{{\lambda_j}^{n-i}}\right]_{1\le i\le n}$.*

Proof. We wish to show $Au_j = \lambda_j u_j$. If the theorem is true, then $\lambda_j u_{j,i} = \lambda_j \frac{1}{{\lambda_j}^{n-i}} = \frac{1}{{\lambda_j}^{n-(i+1)}} = u_{j,i+1}$. Recall the state update (4) of the controllable LDS, which shifts $n-1$ entries and computes a dot product in the last entry:

$$Au_j = \begin{bmatrix} u_{j,2} \\ \vdots \\ u_{j,n-1} \\ -\sum_i a_{i-1}u_{j,i} \end{bmatrix} = \begin{bmatrix} \lambda_j u_{j,1} \\ \vdots \\ \lambda_j u_{j,n} \\ -\sum_i a_{i-1}/{\lambda_j}^{n-i} \end{bmatrix}$$

It suffices to show:

$$-\sum_i a_{i-1}/\lambda_j^{n-i} = \lambda_j u_{j,n} = \lambda_j$$

$$\text{i.e.} \sum_{1\le i\le n} \frac{a_{i-1}}{\lambda_j^{n-(i-1)}} = -1 \tag{9}$$

It is well known that the characteristic polynomial of A is $p(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_{n-1}t^{n-1} + t^n$. By definition, its roots (those t where $p(t) = 0$) are the eigenvalues of A. So each λ_j satisfies:

$$\begin{aligned} 0 =& a_0 + a_1\lambda_j + a_2{\lambda_j}^2 + \ldots + a_{n-1}{\lambda_j}^{n-1} + {\lambda_j}^n \\ =& {\lambda_j}^n \left(1 + \sum_{1\le i\le n} \frac{a_{i-1}}{\lambda_j^{n-(i-1)}}\right) \end{aligned}$$

Either we have a null eigenvalue $\lambda_j = 0$, or we have the desired equation (9). □

Since B is all zero except the last coordinate, $B' = U^{-1}B$ is just the last column of U^{-1}. This can also be expressed in terms of the eigenvalues of A.

Claim 1. *The (unnormalized) last column of U^{-1} is:*

$$B' = \left[\frac{\lambda_i^{n-1}}{\prod_{j\ne i}(\lambda_i - \lambda_j)}\right]_{1\le i\le n} \tag{10}$$

This claim builds upon the previous lemma. Its proof is more involved, so we reserve it for a longer version. The parametrization is summarized below.

Parameters: Vector $C \in \mathbb{R}^{1\times n}$. For $1 \le j \le n/2$, $a_j \in \mathbb{R}$ and $b_j \in \mathbb{R}$.

Algorithm: Define an $n \times n$ matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ (in conjugate pairs) on the diagonal:

$$\Lambda = \begin{bmatrix} a_1 + b_1 i & 0 & 0 & 0 & 0 \\ 0 & a_1 - b_1 i & 0 & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & 0 & a_{n/2} + b_{n/2} i & 0 \\ 0 & 0 & 0 & 0 & a_{n/2} - b_{n/2} i \end{bmatrix}$$

B' is a function of the a_j and b_j defined in (10). For each $1 \le t \le T$, the sequence of states $s'_{t+1} = \Lambda s'_t + B' s_t$ and their gradients $\nabla s'_{t+1}$ (with respect to a_j and b_j) may be computed using the algorithm of proposition 3 on (6). Finally, compute the outputs as $y_t = C' h'_t$, where C' is defined in (6).

Conversion to canonical form (optional): Compute U from Λ via lemma 2. Compute U^{-1} from U via matrix inversion. Compute $A = U\Lambda U^{-1} \in \mathbb{R}^{n\times n}$.

5. Future Work

Since convolution neural networks are so fast and popular, it is somewhat surprising that LDS are not: per equation 2, LDS *are* convolution layers (with an infinite kernel size and only one stride dimension). This work makes reachable, SISO LDS faster. But for them to become as popular as their convolutional brethren, a number of questions still remain to be answered. Just because we *can* train LDS on very long time series doesn't mean we *should*. The vanishing/exploding gradient problem can occur in LDS (Hochreiter, 1998). There is some question about the extent to which RNNs, of any kind, take advantage of long-range dependencies (Miller & Hardt, 2019). Interposing nonlinearities, while retaining the analytic virtues of LDS, remains an open research problem.

References

Blelloch, G. E. Prefix sums and their applications. In *Synthesis of parallel algorithms*, pp. 35—60. Morgan Kaufmann Publishers Inc., 1990. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.6430.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/gehring17a.html.

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

Hardt, M., Ma, T., and Recht, B. Gradient descent learns linear dynamical systems. *arXiv preprint arXiv:1609.05191*, 2016.

Hazan, E., Singh, K., and Zhang, C. Learning linear dynamical systems via spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 6702–6712, 2017.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Huang, W., Harandi, M., Zhang, T., Fan, L., Sun, F., and Huang, J. Efficient optimization for linear dynamical systems with applications to clustering and sparse coding. In *Advances in Neural Information Processing Systems*, pp. 3444–3454, 2017.

Lessard, L., Recht, B., and Packard, A. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Martin, E. and Cundy, C. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyUNwulC-.

Miller, J. and Hardt, M. Stable recurrent models. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hygxb2CqKm.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Recht, B. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*.

Shalit, U. and Chechik, G. Coordinate-descent for learning orthogonal matrices through givens rotations. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 548–556, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/shalit14.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Zhou, K., Doyle, J. C., Glover, K., et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.