Abstract

We present a windowed technique to learn parsimonious time-varying autoregressive models from multivariate timeseries. This unsupervised method uncovers spatiotemporal structure in data via non-smooth and non-convex optimization. In each time window, we assume the data follow a linear model parameterized by a potentially different system matrix, and we model this stack of system matrices as a low rank tensor. Because of its structure, the model is scalable to high-dimensional data and can easily incorporate priors such as smoothness over time. We find the components of the tensor using alternating minimization and prove that any stationary point of this algorithm is a local minimum. In a test case, our method identifies the true rank of a switching linear system in the presence of noise. We illustrate our model’s utility and superior scalability over extant methods when applied to several synthetic and real examples, including a nonlinear dynamical system, worm behavior, sea surface temperature, and monkey brain recordings.

1 Introduction

Data-driven linear models are a natural approach to modeling timeseries and have been studied extensively in the mathematical sciences and applied fields. These domains include Earth and atmospheric sciences [20, 14, 1], fluid dynamics [30, 2, 33], and neuroscience [7, 24]. In nonlinear settings, linear approximations may be justified by their connection to the eigenfunctions of the Koopman operator [8, 23], known as the dynamic mode decomposition [30, 35, 31]. In order to capture non-stationary behavior, it is important to leverage dynamic linear models (DLMs) that vary over time [36]. However, since every linear model is parametrized by a system matrix, and this matrix changes over time, a naive DLM fit to a length $T$ timeseries of $N$ variables depends upon $O(TN^3)$ parameters, which could be very large. We propose to manage such complexity by representing the stack of system matrices as a low rank tensor with only $O(T + 2N)$ parameters (see Fig. 1).

There is a large literature on DLMs, including time-varying autoregressive (TVAR) and switching linear dynamical systems (SLDS) models that we cannot review in full here [36]. Ours is a regularized optimization method [in the spirit of 17, 9, 10, 32], complementary to the Bayesian approaches taken in much of the literature [28, 13, 13, 21]. We also would like to highlight the recent work of Costa et al. [12], which uses likelihood tests to adaptively segment a timeseries and fit different models to each segment. However, our approach is inherently more scalable due to the low-rank assumptions we make.
The key innovation of our model is to parametrize the dynamics by a low rank tensor for computational tractability, ease of identification, and interpretation. Tensor decompositions [19] are a powerful technique for summarizing multivariate data and an area of ongoing research in theory [16, 3, among others] and applications [e.g., to improve neural networks 26, 18]. In our formulation, the system tensor representing the DLM is regressed against the data. In this aspect, our method is most similar to the work of Yu and colleagues who considered spatiotemporal forecasting with spatial smoothness regularization [4, 38, 37], in contrast to our temporal smoothness. Our work also differs in the emphasis on non-smooth regularization to find switching or other temporally structured behavior.

2 The TVART model

We now introduce our time-varying autoregressive model with low rank tensors (TVART, Fig. 1). Assume we have sampled the trajectory of an $N$-dimensional dynamical system $x(t)$ for $t = 1, \ldots, \tau$. We split this trajectory into $T$ non-overlapping windows of length $M$, so that $TM = \tau$. Let $X$ be the $N \times M \times T$ tensor with entries $x_{ijk} = x_i((k-1)M+j)$, and similarly let $Y$ be a tensor of the same size with entries shifted by one time point, $y_{ijk} = x_i((k-1)M+j+1)$.

(See Appendix A for the notation conventions.) We call the frontal slices $X_k$ and $Y_k$ the snapshot matrices for window $k$. The first of these are

$$X_1 = \begin{bmatrix} x(1) & x(2) & \ldots & x(M) \end{bmatrix} \quad \text{and} \quad Y_1 = \begin{bmatrix} x(2) & x(3) & \ldots & x(M+1) \end{bmatrix}.$$  

The subsequent snapshots $X_k, Y_k$ for $k > 1$ are each shifted by $(k-1)M$.

The goal of TVART is to fit an $N \times N \times T$ tensor $A$ of system matrices, so that $Y_k \approx A_k X_k$ for $k = 1, \ldots, T$, where $A_k$ is the $k$th frontal slice of $A$. The assumption underlying this goal is that $x(t+1) \approx A(t) x(t)$ where $A(t)$ is constant within a window. This motivates the least squares optimization problem, equivalent to assuming uncorrelated Gaussian errors,

$$\min_{A: \text{rank}(A) = R} \frac{1}{2} \sum_{k=1}^{T} \| Y_k - A_k X_k \|_F^2 . \quad \text{(TVART)}$$  

Without the rank constraint, TVART factors into decoupled problems for each window $A_k$. In order to limit the degrees of freedom in the tensor $A$, we use a low rank formulation. Specifically, we
represent $A$ using the canonical polyadic (CP) decomposition \cite{19} of rank $R$ as

$$A = \sum_{r=1}^{R} u_r^{(1)} \circ u_r^{(2)} \circ u_r^{(3)}$$

in terms of the factor matrices $U^{(1)} \in \mathbb{R}^{N \times R}$, $U^{(2)} \in \mathbb{R}^{N \times R}$, and $U^{(3)} \in \mathbb{R}^{T \times R}$, where $u_r^{(i)}$ is the $r$th column of $U^{(i)}$. Thus, the number of parameters is reduced to $(2N + T)R$, which is now linear in $N$ and $T$. We can optionally normalize the factors to have unit-length columns and capture their scalings in a vector $\lambda \in \mathbb{R}^R$; we consider this a postprocessing step and explicitly state when we do this.

When solving [TVART] for $A$, we work directly with the frontal slices

$$A_k = U^{(1)} D^{(k)} U^{(2)^T}, \quad \text{where} \quad D^{(k)} = \operatorname{diag}(u^{(3)}_k).$$

Matrix $D^{(k)}$ is the $R \times R$ diagonal matrix formed from the $k$th row of $U^{(3)}$ \cite{19}. We call the matrices $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ the TVART dynamical modes. Specifically, we refer to $U^{(1)}$ and $U^{(2)}$ as the left and right spatial modes, since they determine the loadings of $A_k$ onto the spatial dimensions/channels in the data. The matrix $U^{(3)}$, which determines $D^{(k)}$, contains the temporal modes, since it determines the time-variation of the system matrix $A_k$ across windows.

Equation (1) is similar to the singular value decomposition (SVD); each slice $A_k$ is the product of a low-rank matrix $U^{(1)}$, a diagonal matrix $D^{(k)}$, and another low-rank matrix $U^{(2)^T}$. However, in the SVD, the left and right singular vectors are orthogonal. Let $U^{(1)} = Q^{(1)} R^{(1)}$ and $U^{(2)} = Q^{(2)} R^{(2)}$ be the QR decompositions of the left and right spatial modes, so that $A_k = Q^{(1)} R^{(1)^T} D^{(k)} R^{(2)^T} Q^{(2)^T}$. Thus, in order to calculate the SVD of $A_k$, we would have to take the SVD of the $R \times R$ matrix $R^{(1)} D^{(k)} R^{(2)^T}$. This also illustrates that, in the CP decomposition, slices $A_k$ and $A_{k'}$ may have different left and right singular vectors (but they always lie in the column spaces of $Q^{(1)}$ and $Q^{(2)}$). This flexibility is important to allow fitting of switching models with different singular subspaces.

### 2.1 Extensions: affine dynamics and higher-order autoregressions

In many applications, the mean of the data may drift over time, and thus affine models $x(t+1) = A_k x(t) + b_k$ are more appropriate than linear models. We can fit an affine model of this type within the TVART framework by appending a row of ones to each $X_k$ and extending $U^{(2)}$ by one row to build in a $b_k$ term. In this case, we have that $b_k = U^{(1)} D^{(k)} c$, where $c$ is the extra row $u^{(2)}_{R+1}$.

Furthermore, autoregressive models of higher order are often considered, where $x(t+1)$ is predicted from data with $p$ lags $x(t), \ldots, x(t-p+1)$. In this case, the dimensions of $X$ and $A$ change to $NP \times M \times T$ and $N \times NP \times T$, respectively, but otherwise the mathematics remain equivalent. For simplicity, we focus on just the $p = 1$ case, but higher-order autoregressive models are likely better-suited to certain applications.

### 2.2 Norm regularization

A natural approach to [TVART] is alternating least squares. However, we have found that this is numerically unstable, in particular for a switching linear test problem (Sec. 3.1) with low noise. Additive, independent noise adds a diagonal component to the data covariance, which suggests we apply Tikhonov regularization to the problem. An additional motivation is that we do not want the entries in the matrices $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ to become too large, but some might become large due to the scaling indeterminacy of the CP decomposition \cite{19}. We thus add a Tikhonov regularization term

$$\frac{1}{2\eta} \left( \|U^{(1)}\|_F^2 + \|U^{(2)}\|_F^2 + \|U^{(3)}\|_F^2 \right)$$

to the least-squares loss. The regularization parameter $\eta$ controls the magnitude of this regularization; as $\eta \rightarrow \infty$, the constraint disappears. In matrix completion problems, a similar two-term regularization is often added and can be seen as a convex relaxation of the matrix rank.
We now consider a few possible regularizers on the time components $U^{(3)}$.

### 2.3 Temporal mode smoothing

We now consider a few possible regularizers on the time components $U^{(3)}$. Recall that the rows of $U^{(3)}$ correspond to the loadings at different time windows. By forcing these rows to be correlated, we keep the system matrices $A_k$ from varying too much from window to window, a form of temporal smoothing. The first regularizer we consider is a total variation (TV) penalty:

$$TV(U^{(3)}) = \sum_{r=1}^{R} \sum_{k=2}^{T} |u_{kr}^{(3)} - u_{k-1,r}^{(3)}| = \sum_{r=1}^{R} \|D u_{r}^{(3)}\|_1. \quad (2)$$

Matrix $D$ is the $(T-1) \times T$ first difference matrix (-1 on the principal diagonal and 1 on the lower diagonal). TV prefers piecewise constant time components $u_{kr}^{(3)}$ since it penalizes nonzero first-differences with an $\ell^1$ penalty to enforce sparsity, appropriate for an SLDS. We also consider a spline penalty:

$$\text{Spline}(U^{(3)}) = \frac{1}{2} \sum_{r=1}^{R} \sum_{k=2}^{T} (u_{kr}^{(3)} - u_{k-1,r}^{(3)})^2 = \frac{1}{2} \|DU^{(3)}\|_F^2. \quad (3)$$

This linear smoother penalizes the $\ell^2$-norm of the first derivative, leading to smoothly varying solutions.

### 2.4 Regularized cost function

We modify the problem (TVART), adding the Tikhonov and smoothing penalties to the loss function. These additions result in the regularized cost function

$$C = \frac{1}{2} \sum_{k=1}^{T} \|Y_k - A_k X_k\|_F^2 + \frac{1}{2\eta} \left( \|U^{(1)}\|_F^2 + \|U^{(2)}\|_F^2 + \|U^{(3)}\|_F^2 \right) + \beta \mathcal{R}(U^{(3)}), \quad (4)$$

where $A_k$ follows equation (1) as before. Here, $\mathcal{R}(\cdot)$ is either TV($\cdot$) or Spline($\cdot$). Increasing the temporal smoothing strength $\beta$ leads to stronger regularization, as does decreasing $\eta$.

### 2.5 Alternating minimization algorithm

We minimize the regularized cost (4) using alternating minimization, also known as block coordinate descent. In Algorithm 1 we give the full alternating minimization procedure for solving the regularized problem. The subroutines that minimize for $U^{(1)}$, $U^{(2)}$ and $U^{(3)}$ require different approaches. Since the objective (4) is quadratic in $U^{(1)}$ and $U^{(2)}$, we find these by solving a linear matrix equation either directly or using the method of conjugate gradients (CG); this works best for solving the Sylvester equation in $U^{(2)}$. For $U^{(3)}$, with the Spline penalty, the cost is again quadratic so we also use CG. However, the TV penalty is convex but not smooth, so in this case we use the proximal gradient method with Nesterov acceleration. See Appendix B for further algorithm details. Our code is available from https://github.com/kharris/tvart.

**Theorem 2.1.** The sequence of iterates generated by Algorithm 1 is defined, bounded, and every cluster point is a coordinatewise minimum (Nash point) of the regularized cost (4).

**Proof.** We use the framework of Tseng [34, Theorem 5.1] for cyclic block coordinate descent (of which Algorithm 1 is an example) on objective functions with convex and lower semicontinuous...
blocks, as well as bounded level sets. We split the cost into smooth and non-smooth parts
\[ C(U^{(1)}, U^{(2)}, U^{(3)}) = f_0(U^{(1)}, U^{(2)}, U^{(3)}) + f_1(U^{(3)}), \]
where \( f_1(U^{(3)}) = \beta R(U^{(3)}) \) and \( f_0 \) contains the remaining loss and Tikhonov terms. The function \( f_0 \) is continuous and differentiable, and it is \( \frac{1}{\beta} \)-strongly convex in each of its blocks \( U^{(1)}, U^{(2)}, \) and \( U^{(3)} \). However, \( f_0 \) is not a convex function. Also, \( f_1 \) is convex and continuous. Let \( (U^{(1)}_0, U^{(2)}_0, U^{(3)}_0) \) be the initialization and \( a = C(U^{(1)}_0, U^{(2)}_0, U^{(3)}_0) \). Denote the level set
\[ S_a = \{(U^{(1)}, U^{(2)}, U^{(3)}): C(U^{(1)}, U^{(2)}, U^{(3)}) \leq a\}. \]
Then, since \( C(U^{(1)}, U^{(2)}, U^{(3)}) \geq \frac{1}{\beta} \left( ||U^{(1)}||_F^2 + ||U^{(2)}||_F^2 + ||U^{(3)}||_F^2 \right) \) and the ball \( B_0(r) = \{x: ||U^{(1)}||_F^2 + ||U^{(2)}||_F^2 + ||U^{(3)}||_F^2 \leq r\} \) is bounded, we can conclude that the level set \( S_a \subseteq B_0(2\alpha) \) is also bounded. Then by Tseng [34, Theorem 5.1], we obtain the result.

**Remark.** We did not use any structure of \( R \) besides convexity and lower semicontinuity, thus the same convergence results hold for other regularizations with those properties. However, we did need to use the Tikhonov penalty to ensure that the level sets are bounded.

3 Example applications

In this section we test our method on both synthetic data (switching linear and nonlinear), as well as real-world datasets. With the synthetic linear data, we show that our algorithm can recover the true dynamics and is competitive with other state of the art techniques. In the other examples, we highlight how the recovered modes are interpretable and can correspond to important dynamical regimes. For detailed explanation of the datasets and parameters used, please refer to Appendix D.

3.1 Test problem: switching low rank linear system

We first apply TVART to a simple test case where the true model is a low-rank, switching linear system. We generate two \( N \times N \) system matrices, \( A_1 \) and \( A_2 \), which are random, rank-2 rotation matrices. For the first half of our timeseries, the dynamics follow \( A_1 \), and then they switch to \( A_2 \). Gaussian noise is added to each entry of \( x(t) \) to form the observations.
In Fig. 2 we see that, across system sizes $N$, TV ART is able to recover the true dynamics $A$ as well as or better than a rank 4 SLDS. The rank 6 SLDS performs close to independent rank 4, while the independent models of higher rank perform worse. On the right of Fig. 3 the runtimes of rank 4 TV ART and SLDS are compared. We find that TV ART offers a speedup of approximately 3x–6x over SLDS for $N \geq 10^3$, and the runtimes of both appear to scale superlinearly as $\sim N^{1.8}$. We conclude that TV ART with TV regularization is a scalable, alternative way of finding switching linear dynamics.

A further comparison to smoothly-varying low rank dynamics is given in the Appendix D.1. In this case, TV ART with Spline regularization performs much better than SLDS, since it can capture smooth behavior which SLDS cannot.

### 3.2 Nonlinear example: Lorenz (1963) system

We take a sample trajectory of the canonical Lorenz [22] chaotic nonlinear dynamical system, a three variable dynamical system that exhibits nonlinear oscillations about two unstable spiral points. The oscillations grow about a single fixed point until the system “flips” to oscillation about the other fixed point. There is a third unstable node at the origin.

In Fig. 4 we depict the temporal modes as well as the result of clustering the $A_k$ matrices using k-medoids algorithm on $U^{(3)}$ with 3 clusters. The clustering is able to identify the lobe of the attractor that the system is on (clusters 2 and 3) as well as if it is in a transitory state between the two lobes.
Figure 4: TV ART applied to Lorenz (1963) system data with parameters as given in the text. We show the observations from the Lorenz system as well as the temporal modes. The temporal modes are able to track the reversals where the first two state variables change sign. Clustering the temporal modes reveals the three dominant dynamical regimes.

Figure 5: TVART applied to C. elegans worm dataset. In this example, the worm begins moving forward, executes a turn, and continues but in the backward direction. The temporal modes in TVART are able to pick out these three regimes, and a clustering of them identifies these three behavioral states.

(cluster 1). This makes sense because in each of these cases the dynamics are dominated by the closest fixed point.

3.3 Worm behavior

We analyze the escape response behavior of the nematode worm *Caenorhabditis elegans* in response to a heat stimulus [5, 6]. These were used as test data for clustering via adaptive linear models in the recent work of Costa et al. [12]. The results for worm 1 are shown in Fig. 5. We performed clustering on the system matrices as before with three clusters and found that these clusters matched the three behaviors in the data: forward crawling, a turn, and backward crawling. These clusters are not trivial: the data means during forward and backward motion are approximately equal, but the phase velocity switches.

Finally, we also compared our results to the code provided by Costa et al. [12], which fits an adaptive linear model and clusters the same timeseries; the clustering results are essentially the same. In terms of runtime, fitting a TV ART model, performing clustering, and displaying the results takes 23 s (90 iterates), versus 123 s for the code of Costa et al. [12]. Thus, we see that our method is much faster than theirs, while producing essentially the same clustering results.

3.4 Sea surface temperature

As an example of a high-dimensional dataset, we applied our method to weekly sea surface temperature data from 1990 until present [29]. The leading modes that are output by the algorithm oscillate seasonally. However, we also find a temporal mode that tracks the El Niño-Southern Oscillation (ENSO), Fig. 6. The corresponding spatial modes show a plume of warm water in the central and eastern equatorial Pacific Ocean. Warmer than average water in this location is the signature of ENSO.
3.5 Neural activity during a reaching task

A second high-dimensional dataset we use comes from electrocorticography recordings (ECoG, measured with chronic electrodes placed below the skull and the dura mater) of a Japanese macaque monkey *Macaca fuscata* during a reaching task [11]. During the task, the monkey makes repeated reaches towards food while its limbs are tracked using motion capture and brain activity is recorded. Figure 7 depicts the results: We show motion capture data with dashed lines at the onset and offsets of movements, using a changepoint detection procedure. We also highlight the result of clustering the TVART temporal modes into two clusters. The brain activity reveals two dominant modes, one of which is aligned to movement. These movements are accompanied by an increase in high frequency (32-200 Hz) and a decrease in low frequency (2-32 Hz) power. The dominant TVART mode (show in blue) follows this spectral change in the timeseries, and the left spatial mode associated with it is centered in the premotor region (supplementary Fig. 9). TVART tracks this spectral change directly from the timeseries of electrode voltage.

4 Conclusions

We have presented a low rank tensor formulation of time-varying autoregressive problems that we call TVART. This offers the advantage of being able to scale to problems with many state variables or many time points, as highlighted by our examples. Furthermore, the modes output by our method are often interpretable as dominant dynamical features in the data. Future work should investigate higher-order autoregressive models and applications to other datasets such as economic data.

Acknowledgements

Thank you to Scott Linderman and Nathan Kutz for discussions and to Steven Peterson for help preprocessing the Neurotycho data. KDH was supported by a Washington Research Foundation Postdoctoral Fellowship. BWB, RR, and KDH were supported by NSF CNS award 1630178 and DARPA award FA8750-18-2-025.
[27] Perraudin, N., Kalofolias, V., Shuman, D., and Vandergheynst, P. (2014). UNLocBoX: A MATLAB convex optimization toolbox for proximal-splitting methods. arXiv:1402.0779 [cs, stat].

[28] Prado, R., Huerta, G., and West, M. (2000). Bayesian time-varying autoregressions: Theory, methods and applications. Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo, 4(4):405–422.

[29] Reynolds, R. W., Rayner, N. A., Smith, T. M., Stokes, D. C., and Wang, W. (2002). An Improved In Situ and Satellite SST Analysis for Climate. Journal of Climate, 15(13):1609–1625.

[30] Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P., and Henningson, D. S. (2009). Spectral analysis of nonlinear flows. Journal of Fluid Mechanics, 641:115–127.

[31] Takeishi, N., Kawahara, Y., and Yairi, T. (2017). Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems 30, pages 1130–1140. Curran Associates, Inc.

[32] Tank, A., Fox, E. B., and Shojaie, A. (2017). An Efficient ADMM Algorithm for Structural Break Detection in Multivariate Time Series. arXiv:1711.08392 [stat].

[33] Towne, A., Schmidt, O. T., and Colonius, T. (2018). Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. Journal of Fluid Mechanics, 847:821–867.

[34] Tseng, P. (2001). Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. Journal of Optimization Theory and Applications, 109(3):475–494.

[35] Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., and Kutz, J. N. (2014). On dynamic mode decomposition: Theory and applications. Journal of Computational Dynamics, 1(2):391–421.

[36] West, M. and Harrison, J. (1997). Bayesian Forecasting and Dynamic Models. Springer Science & Business Media.

[37] Yu, R., Li, G., and Liu, Y. (2018). Tensor Regression Meets Gaussian Processes. In International Conference on Artificial Intelligence and Statistics, pages 482–490.

[38] Yu, R. and Liu, Y. (2016). Learning from Multiway Data: Simple and Efficient Tensor Regression. In International Conference on Machine Learning, pages 373–381.
Supplemental Appendix
“Time-varying Autoregression with Low Rank Tensors”

A Notation

Throughout this work, we follow the notational conventions of Kolda and Bader [19]. In general, tensors are denoted with calligraphic bold (A), matrices with capital bold (A), and vectors with lower-case bold symbols (a). We use MATLAB style “colon” notation to represent slices, e.g. A, is the row vector formed from the ith row of a matrix A. All of the tensors we consider are 3rd-order. For a tensor A, let A, be its kth frontal slice, i.e. A,.

B Alternating least squares algorithm

Here we describe in detail how to optimize the loss of the unregularized (TV ART) problem. The computations required for the regularized problem are essentially the same, with added diagonal matrices to satisfy the Tikhonov assumptions (not shown). Let the loss function be

\[ L(U^{(1)}, U^{(2)}, U^{(3)}) = \frac{1}{2} \sum_{k=1}^{T} \| Y_k - U^{(1)} D^{(k)} U^{(2)\top} X_k \|_F^2. \]  

(B.1)

Then minimizing the loss is an equivalent formulation of the TV ART problem. The cost (B.1) is quadratic and convex in each of the variables U, U, and U, although it is not jointly convex in all of them. Smooth and convex least-squares problems are some of the most studied in optimization, leading to linear equations for the unknowns. In the next three sections, we explicitly lay out the gradients of L with respect to each mode which are used in Algorithm 1.

B.1 Subproblem 1: Left Spatial Modes U

Taking partial derivatives of L with respect to U leads to

\[ \nabla_{U^{(1)}} L = U^{(1)} \left( \sum_{k=1}^{T} D^{(k)} U^{(2)\top} X_k X_k\top U^{(2)} D^{(k)} \right) - \left( \sum_{k=1}^{T} Y_k X_k\top U^{(2)} D^{(k)} \right). \]  

(B.2)

Solving \( \nabla_{U^{(1)}} L = 0 \) requires forming an \( N \times R \) matrix, and right multiplying by the pseudoinverse of an \( R \times R \) matrix.

B.2 Subproblem 2: Right Spatial Modes U

We now differentiate L with respect to U and obtain

\[ \nabla_{U^{(2)}} L = \sum_{k=1}^{T} X_k X_k\top U^{(2)} D^{(k)} U^{(1)\top} U^{(1)} D^{(k)} - \sum_{k=1}^{T} X_k Y_k\top U^{(1)} D^{(k)}, \]  

(B.3)

which we can rewrite in the form

\[ \nabla_{U^{(2)}} L = \sum_{k=1}^{T} L_k U^{(2)} R_k - B \]  

(B.4)

where \( B \in \mathbb{R}^{N \times R} \), \( L_k \in \mathbb{R}^{N \times N} \), and \( R_k \in \mathbb{R}^{R \times R} \).

For small enough system sizes, we can use the Kronecker product \( \otimes \) and vectorization operations to rewrite (B.4) as a linear matrix-vector equation

\[ \text{vec}(\nabla_{U^{(2)}} L) = \left( \sum_{k=1}^{T} R_k \otimes L_k \right) \text{vec}(U^{(2)}) - \text{vec}(B). \]  

(B.5)

If we set the derivative equal to zero, we obtain a square \( NR \times NR \) linear system in \( NR \) unknowns. However, for large \( N \), it is impractical to even form this matrix. Thus, we solve this via the matrix-valued method of conjugate gradients.
B.3 Subproblem 3: Temporal Modes $U^{(3)}$

Finally, we derive the gradients of $L$ with respect to $U^{(3)}$. The equations for $U^{(3)}$, that is for $D^{(k)}$, are similar except in this case the unknown is a diagonal matrix. The gradients are decoupled across windows (frontal slices), thus for the $k$th window we have

$$\nabla_{D^{(k)}} L = \left( (U^{(2)})^T X_k^T X_k^T U^{(2)} D^{(k)} U^{(1)} - (U^{(2)})^T X_k Y_k^T U^{(1)} \right) * I,$$  

(2.6)

where $*$ is the Hadamard elementwise product, which enforces the constraint that the gradient with respect to $D^{(k)}$ is also diagonal.

Define the $R \times R$ matrices $L_k = (U^{(2)})^T X_k X_k^T U^{(2)}$ and $R_k = (U^{(1)})^T U^{(1)}$. Then by Theorem 2.5 in Million [25] we have that

$$(L_k' D^{(k)} R_k')_{ii} = \left( (L_k' * R_k') \left( u_{k,c}^{(3)} \right)^T \right)_{ii},$$

where we have substituted $D^{(k)} = \text{diag}(u_{k,c}^{(3)})$. Thus, the gradient can be written as

$$\text{vecdiag}(\nabla_{D^{(k)}} L) = \left( L_k' * R_k' \left( u_{k,c}^{(3)} \right)^T - \text{vecdiag} \left( (U^{(2)})^T X_k Y_k^T U^{(1)} \right) \right),$$

where $\text{vecdiag}(M)$ is the column vector formed by the diagonal elements of $M$. Thus, the least-squares problem for $U^{(3)}$ involves solving an $R \times R$ system for each row $u_{k,c}^{(3)}$ of $U^{(3)}$ independently, $k = 1, \ldots, T$. However, adding regularization to the temporal modes destroys this decoupling across time windows, which is why we resort to CG or proximal gradient approaches in Algorithm 1.

B.4 Computational complexity of alternating least squares

Each minimization step requires the solution of a linear matrix equation $\nabla_{U^{(d)}} L = 0$ for $U^{(d)}$. We now comment on the difficulty of solving the subproblems in $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$. Subproblem 1 in $U^{(1)}$ is by far the easiest, since it involves just one $R \times R$ system solve or pseudoinverse. Subproblem 3 in $U^{(3)}$ requires $T$ decoupled $R \times R$ system solves. The cost of this step is thus linear in $T$. We assume that $R$ is small enough that solving for each time window is quick. By far the most difficult step is Subproblem 2 for the right hand factors $U^{(2)}$. Partly this is because we obtain a Sylvester equation which we might naively solve as an $NR \times NR$ square system in $NR$ unknowns. In practice, we avoid using Kronecker products and instead use a matrix-free method such as conjugate gradients, we can expect this to take less memory and possibly less time, depending on the condition number. However, in either case the subproblem involves coefficient matrices of size $N$, whereas for the other problems these are of size $R$.

The full alternating minimization procedure for the fully regularized problem, Algorithm 1, has similar complexity. However, in that case the cost is dominated by the minimization over $U^{(3)}$, which requires computing the proximal operator at each step of the proximal gradient method.

B.5 Gradient Identities for Least Squares Subproblems

The least squares subproblems for $U^{(2)}$ and $U^{(3)}$ require us to compute a gradient of a particular form:

$$\frac{\partial}{\partial B} \frac{1}{2} \| Y - AB^T C \|^2_F = \frac{1}{2} \text{Tr} \left[ (Y - AB^T C) (Y^T - C^T BA^T) \right] = -CY^T A + CC^T B A^T.$$

(2.7)

Replacing $Y \rightarrow Y_k, A \rightarrow U^{(1)} D^{(k)} C \rightarrow X_k, and B \rightarrow U^{(2)}$ leads to the terms in the Sylvester equation (2.3) for the unknown $U^{(2)}$. On the other hand, replacing $Y \rightarrow Y_k, A \rightarrow U^{(1)} C \rightarrow U^{(2)} X_k, and B \rightarrow D^{(k)}$ and taking the Hadamard product with the identity matrix (i.e., enforcing the diagonal constraint) leads to the terms in (2.6).

C Algorithm implementation details

The code and instructions for running it are available from https://github.com/kharris/tvart. We implemented Algorithm 1 in MATLAB. It was run on an Intel(R) Xeon(R) CPU E5-2620v4 @ 2.10GHz, 1200 MHz with 32 cores and 128 GB RAM using Ubuntu 16.04.1 with Linux.
Table 1: Dataset parameters

| Name                     | N     | M   | T    | R   | η   | β   | R   | affine? |
|--------------------------|-------|-----|------|-----|-----|-----|-----|---------|
| Switching test problem   | 6–4000| 20  | 10   | 4, 6, 8 | N⁻¹ | 5   | TV  | no      |
| Lorenz system            | 3     | 10  | 419  | 4   | 0.01| 400 | TV  | yes     |
| Worm behavior            | 4     | 6   | 33   | 6   | 0.05| 6   | TV  | yes     |
| Sea surface temperature  | 1259  | 9   | 171  | 6   | 10⁻³| 10⁴ | Spline | no     |
| Neural activity          | 64    | 200 | 1999 | 8   | 1   | 100 | TV  | no      |

kernel 4.15.0-48-generic and 64-bit MATLAB R2017b. The CG and prox-gradient subroutines are limited to 24 and 40 iterates, respectively. Proximal gradient uses a backtracking line search to find the step size and the proximal operator of TV is evaluating using prox\_tv1d from UNLocBox [27]; this is typically the bottleneck step for large problems. All hyperparameter tuning (for R, M, η, and β) was performed manually.

The initialization is the following method: We consider the entire timeseries in two snapshot matrices,

\[
X = \begin{bmatrix} x(1) & x(2) & \ldots & x(TM) \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} x(2) & x(3) & \ldots & x(TM+1) \end{bmatrix}.
\]

We then fit a single model to the timeseries by \[A = YX^\dagger\] and form the matrices \([U, S, V] = \text{svd}(A)\). When \(R = N\), the TV ART modes are then initialized to \(U_0^{(1)} = U, U_0^{(2)} = V\), and \(U_0^{(3)} = 11T/\sqrt{T}\) is the matrix of all-ones with columns normalized. If \(R < N\), we truncate the smaller singular vectors, and when \(R > N\), we add columns equal to \(1/\sqrt{N}\) to \(U_0^{(1)}\) and \(U_0^{(2)}\) and \(1/\sqrt{T}\) to \(U_0^{(3)}\). Initializations with unequal columns appear to help speed up the initial phase of the optimization, as opposed to all-ones. For this reason, we add Gaussian noise to the initializations with standard deviation \(0.5/\sqrt{N}\) to \(U_0^{(1)}\) and \(U_0^{(2)}\) and with standard deviation \(0.5/\sqrt{T}\) to \(U_0^{(3)}\). Initializing to zeros is not appropriate since the gradients in that case are always zero. All of the results we present do not depend strongly on the realization of the noise.

For stopping criteria, we use both a relative and absolute tolerance on the decrease in the cost function. Let \(c(t)\) be the cost \(C(4)\) at iterate \(t\). Then, we stop if either

\[
\frac{|c(t) - c(t - 1)|}{c(t - 1)} < \text{rtol} \quad \text{or} \quad |c(t) - c(t - 1)| < \text{atol}.
\]

Unless specified otherwise, \(\text{rtol} = 10^{-4}\) and \(\text{atol} = 10^{-6}\). In all cases we have tested, the relative tolerance is achieved first. We report the cost \(C(4)\) as it runs as well as the root mean square error (RMSE), which we define as:

\[
\text{RMSE} = \sqrt{\frac{1}{NMT} \sum_{k=1}^{T} \|Y_k - A_kX_k\|_F^2}.
\]

The RMSE is normalized so that it gives a measure of average one-step prediction error per channel. This “goodness of fit” metric can then be compared to the standard deviation of the data.

D Example application details

In this section, we provide precise details of the test problem and datasets we considered. A complete table of parameters is given in Table 1.

D.1 Test problem

We generate the matrices \(A_i\) by the following process for \(i = 1, 2\):

1. Form a \(2 \times 2\) rotation matrix \(A_i' = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix}\).
In Fig. 3, we sweep across the parameter space of the rank 2 orthogonal dynamics that are spanned by the first $R = 6$ principal components. The SLDS$(R)$ models fit the same data with the independent models fit a matrix independently to each window of size $N = 60$ log $\beta$, indep($R$) first performs an SVD on the data and fit the model in the subspace spanned by the first $R$ principal components. The SLDS($R$) models fit the same data with the TV ART algorithm provided by Scott Linderman at [https://github.com/slinderman/ssm](https://github.com/slinderman/ssm). In all cases the maximum number of switches $\kappa_{\max} = 4$. The SLDS model was fit using the mean field variational posterior for 6000 iterations, and the predicted system matrix $A(t)$ was taken as the maximum a posteriori estimate. For the runtime calculation of SLDS, we:

1. First fit the SLDS for 6000 iterations,
2. Took the evidence lower bound (ELBO) sequence and smoothed it using a 2nd-order low-pass Butterworth filter with cutoff frequency 0.01 per iteration,
3. Fit a sigmoid to the smoothed ELBO, and
4. Chose the final iteration number as the estimated step at which the ELBO reaches 99.9% of its maximum according to the sigmoid.

The SLDS was then refit for that number of iterations to compute the run time.

Another comparison was made by assuming rank 2 orthogonal dynamics that are smooth. However, rather than switching the angle $\theta_1$ once, we sample $\bar{\theta}(t)$ from a Gaussian process in order to have slowly-varying rank 2 dynamics. In this case, the orthogonal projection into the higher space $W$ is fixed for all time. The results are shown in Figure [8]. We see that TVART(4) highly outperforms SLDS(6) (fit in the same manner as for the switching test problem described previously) as well as the independent models across system sizes $N$. The parameters used for the sweep were $N = 6, 12, 24, 50, 100, 200, 400, 1000, 2000, 4000$, rank $R = 4$, $M = 1$, $\eta = 6/N$, and $\beta = 600 \log^2_{10}(N)$ with the Spline regularization. Interestingly, with this strong temporal regularization, we can fit a different model at each time point.

### D.2 Lorenz system

We use the usual chaotic parameter set $\sigma = 10$, $r = 28$, and $b = 8/3$, integrate with 4th order Runge-Kutta using step size 0.001, and ignore the initial 1000 step transient. We observe all three state variables corrupted with additive Gaussian noise with unit standard deviation.

We run TVART with $R = 4$, $M = 10$, $\eta = 0.01$, $\beta = 400$, and affine dynamics. The algorithm converges in 369 iterations to an RMSE of 1.39, which is significantly smaller than the standard deviations of the state variables (7.83, 8.98, 8.76 for $x_1, x_2, x_3$, respectively), meaning that the learned TVART model has reasonable one-step predictive power.
Figure 8: Smoothly-varying linear test case. (Above) State variable observations and temporal modes for an example fit with $N = 10$. (Below) Inference error of TV ART(4) versus independent, rank-truncated, and SLDS(6) models as measured in operator and max norms. The max norm difference decreases to zero because the entries in the matrix decrease as $N^{-1/2}$.

D.3 Worm behavior

Worm postural data were analyzed as smooth timeseries of $N = 4$ “eigenworm” principal components. We ran TVART with an affine model and $R = 6, M = 6, \eta = 0.05$, and $\beta = 6$ on these data. We also compared the performance of the code provided by Costa et al. [12] at https://github.com/AntonioCCosta/local-linear-segmentation.

D.4 Sea surface temperature

Sea surface temperature grids [29] were downsampled by a factor of 6 in the latitudinal and longitudinal directions, resulting in final vectors of length $N = 1259$. We chose a window size of $M = 9$ weeks, resulting in $T = 171$ windows, and use parameters $\eta = 10^{-3}, \beta = 10^4$, and the Spline regularizer. In this case, the ALS routine stagnates (the matrix $U^{(2)}$ converges very slowly).
However, we have found that restarting the algorithm after 10 iterations and setting $U^{(2)} = U^{(1)}$ in the new initialization speeds things up significantly. This heuristic was inspired by the fact that the solution has $U^{(2)} \approx U^{(1)}$. In the end, the algorithm requires 1176 iterations to converge.

D.5 Neural activity during a reaching task

The ECoG recordings were provided by Chao et al. [11] as part of the NeuroTycho project. We analyzed the data of monkey K1, date 2009-05-25. We ran TVART on the $N = 64$ channel ECoG voltage data after filtering out 50 Hz line noise, downsampling to 500 Hz, and standardization. The parameters were $M = 200$, $R = 8$, $\eta = 1$, and $\beta = 100$, resulting in $T = 1999$ windows of length 0.4 s.

Figure 9: The dominant spatial modes (left and right) of TVART applied to the neural activity dataset. In the left mode, the electrodes with largest weight are centered in the premotor region known to be active during reaching, posterior of the arcuate sulcus and more medial than lateral. The right spatial mode is less interpretable.