Supplementary Materials for: Switching Autoregressive Low-rank Tensor Models

Table of Contents

A SALT Optimization via Tensor Regression

Let $\mathbf{y}_t \in \mathbb{R}^{N_1}$ be the t-th outputs and $\mathbf{X}_t \in \mathbb{R}^{N_2 \times N_3}$ be the t-th inputs. The regression weights are a tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, which we model via a Tucker decomposition,

$$\mathcal{A} = \sum_{i=1}^{D_1} \sum_{j=1}^{D_2} \sum_{k=1}^{D_3} g_{ijk} \, \mathbf{u}_{:i} \circ \mathbf{v}_{:j} \circ \mathbf{w}_{:k}, \tag{10}$$

where $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{w}_k$ are columns of the factor matrices $\mathbf{U} \in \mathbb{R}^{N_1 \times D_1}$, $\mathbf{V} \in \mathbb{R}^{N_2 \times D_2}$, and $\mathbf{W} \in \mathbb{R}^{N_3 \times D_3}$, respectively, and g_{ijk} are entries in the core tensor $\mathcal{G} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$.

Consider the linear model, $\mathbf{y}_t \sim \mathcal{N}(\mathcal{A} \times_{2,3} \mathbf{X}_t, \mathbf{Q})$ where $\mathcal{A} \times_{2,3} \mathbf{X}_t$ is defined using the Tucker decomposition of $\mathcal{A}$ as,

$$\begin{aligned} \mathcal{A} \times_{2,3} \mathbf{X}_t &= \mathcal{A}_{(1)} \mathrm{vec}(\mathbf{X}_t) && (11)\\ &= \mathbf{U} \mathcal{G}_{(1)} (\mathbf{V}^\top \otimes \mathbf{W}^\top) \mathrm{vec}(\mathbf{X}_t) && (12)\\ &= \mathbf{U} \mathcal{G}_{(1)} \mathrm{vec}(\mathbf{V}^\top \mathbf{X}_t \mathbf{W}) && (13) \end{aligned}$$

where $\mathcal{A}_{(1)} \in \mathbb{R}^{N_1 \times N_2 N_3}$ and $\mathcal{G}_{(1)} \in \mathbb{R}^{D_1 \times D_2 D_3}$ are mode-1 matricizations of the corresponding tensors. *Note that these equations assume that matricization and vectorization are performed in row-major order, as in Python but opposite to what is typically used in Wikipedia articles.*

Equation (13) can be written in multiple ways, and these equivalent forms will be useful for deriving the updates below. We have,

$$\begin{aligned} \mathcal{A} \times_{2,3} \mathbf{X}_t &= \mathbf{U} \mathcal{G}_{(1)} (\mathbf{I}_{D_2} \otimes \mathbf{W}^\top \mathbf{X}_t^\top) \mathrm{vec}(\mathbf{V}^\top) && (14)\\ &= \mathbf{U} \mathcal{G}_{(1)} (\mathbf{V}^\top \mathbf{X}_t \otimes \mathbf{I}_{D_3}) \mathrm{vec}(\mathbf{W}) && (15)\\ &= \left[\mathbf{U} \otimes \mathrm{vec}(\mathbf{V}^\top \mathbf{X}_t \mathbf{W})\right] \mathrm{vec}(\mathcal{G}). && (16) \end{aligned}$$

We minimize the negative log likelihood by coordinate descent.

Optimizing the output factors Let

$$\widetilde{\mathbf{x}}_t = \mathcal{G}_{(1)} \mathrm{vec}(\mathbf{V}^\top \mathbf{X}_t \mathbf{W}) \tag{17}$$

for fixed $\mathbf{V}$, $\mathbf{W}$, and $\mathcal{G}$. The NLL as a function of $\mathbf{U}$ is,

$$\mathcal{L}(\mathbf{U}) = \frac{1}{2} \sum_t (\mathbf{y}_t - \mathbf{U}\widetilde{\mathbf{x}}_t)^\top \mathbf{Q}^{-1} (\mathbf{y}_t - \mathbf{U}\widetilde{\mathbf{x}}_t). \tag{18}$$

This is a standard least squares problem with solution

$$\mathbf{U}^\star = \left(\sum_t \mathbf{y}_t \widetilde{\mathbf{x}}_t^\top\right) \left(\sum_t \widetilde{\mathbf{x}}_t \widetilde{\mathbf{x}}_t^\top\right)^{-1}. \tag{19}$$

Optimizing the core tensors Let $\widetilde{\mathbf{X}}_t = \mathbf{U} \otimes \mathrm{vec}(\mathbf{V}^\top \mathbf{X}_t \mathbf{W}) \in \mathbb{R}^{N_1 \times D_1 D_2 D_3}$ denote the coefficient on $\mathrm{vec}(\mathcal{G})$ in eq. (16). The NLL as a function of $\mathbf{g} = \mathrm{vec}(\mathcal{G})$ is,

$$\mathcal{L}(\mathbf{g}) = \frac{1}{2} \sum_t (\mathbf{y}_t - \widetilde{\mathbf{X}}_t \mathbf{g})^\top \mathbf{Q}^{-1} (\mathbf{y}_t - \widetilde{\mathbf{X}}_t \mathbf{g}). \tag{20}$$

The minimizer of this quadratic form is,

$$\mathbf{g}^\star = \left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1} \widetilde{\mathbf{X}}_t\right)^{-1} \left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1} \mathbf{y}_t\right) \tag{21}$$

Optimizing the input factors Let

$$\widetilde{\mathbf{X}}_t = \mathbf{U}\mathcal{G}_{(1)}(\mathbf{I}_{D_2} \otimes \mathbf{W}^\top \mathbf{X}_t^\top) \tag{22}$$

for fixed $\mathbf{U}$, $\mathbf{W}$, and $\mathcal{G}$. The NLL as a function of $\mathbf{v} = \mathrm{vec}(\mathbf{V}^\top)$ is,

$$\mathcal{L}(\mathbf{v}) = \frac{1}{2}\sum_t (\mathbf{y}_t - \widetilde{\mathbf{X}}_t\mathbf{v})^\top \mathbf{Q}^{-1}(\mathbf{y}_t - \widetilde{\mathbf{X}}_t\mathbf{v}). \tag{23}$$

The minimizer of this quadratic form is,

$$\mathbf{v}^\star = \left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1}\widetilde{\mathbf{X}}_t\right)^{-1}\left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1}\mathbf{y}_t\right) \tag{24}$$

Optimizing the lag factors Let

$$\widetilde{\mathbf{X}}_t = \mathbf{U}\mathcal{G}_{(1)}(\mathbf{V}^\top \mathbf{X}_t \otimes \mathbf{I}_{D_3}) \tag{25}$$

for fixed $\mathbf{U}$, $\mathbf{V}$, and $\mathcal{G}$. The NLL as a function of $\mathbf{w} = \mathrm{vec}(\mathbf{W})$ is,

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_t (\mathbf{y}_t - \widetilde{\mathbf{X}}_t\mathbf{w})^\top \mathbf{Q}^{-1}(\mathbf{y}_t - \widetilde{\mathbf{X}}_t\mathbf{w}). \tag{26}$$

The minimizer of this quadratic form is,

$$\mathbf{w}^\star = \left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1}\widetilde{\mathbf{X}}_t\right)^{-1}\left(\sum_t \widetilde{\mathbf{X}}_t^\top \mathbf{Q}^{-1}\mathbf{y}_t\right) \tag{27}$$

Multiple discrete states If we have discrete states $z_t \in \{1, \dots, H\}$ and each state has its own parameters $(\mathcal{G}^{(h)}, \mathbf{U}^{(h)}, \mathbf{V}^{(h)}, \mathbf{W}^{(h)}, \mathbf{Q}^{(h)})$, then letting $\omega_t^{(h)} = \mathbb{E}[z_t = h]$ denote the weights from the E-step, the summations in coordinate updates are weighted by $\omega_t^{(h)}$. For example, the coordinate update for the core tensors becomes,

$$\mathbf{g}^{(h)\star} = \left(\sum_t \omega_t^{(h)}\widetilde{\mathbf{X}}_t^{(h)\top}\mathbf{Q}^{(h)-1}\widetilde{\mathbf{X}}_t^{(h)}\right)^{-1}\left(\sum_t \omega_t^{(h)}\widetilde{\mathbf{X}}_t^{(h)\top}\mathbf{Q}^{(h)-1}\mathbf{y}_t\right) \tag{28}$$

B SALT approximates a (Switching) Linear Dynamical System

We now re-state and provide a full proof for Proposition 1.

Proposition 1 (Low-Rank Tensor Autoregressions Approximate Stable Linear Dynamical Systems)**.** *Consider a stable linear time-invariant Gaussian dynamical system. We define the steady-state Kalman gain matrix as* $\mathbf{K} = \lim_{t\to\infty} \mathbf{K}_t$*, and* $\boldsymbol{\Gamma} = \mathbf{A}(\mathbf{I} - \mathbf{K}\mathbf{C})$*. The matrix* $\boldsymbol{\Gamma} \in \mathbb{R}^{D\times D}$ *has eigenvalues* $\lambda_1, \ldots, \lambda_D$*. Let* $\lambda_{\mathsf{max}} = \max_d |\lambda_d|$*; for a stable LDS,* $\lambda_{\mathsf{max}} < 1$ [22]*. Let* n *denote the number of real eigenvalues and* m *the number of complex conjugate pairs. Let* $\hat{\mathbf{y}}_t^{(\mathsf{LDS})} = \mathbb{E}[\mathbf{y}_t \mid \mathbf{y}_{<t}]$ *denote the predictive mean under a steady-state LDS, and* $\hat{\mathbf{y}}_t^{(\mathsf{SALT})}$ *the predictive mean under a SALT model. An order-*L *Tucker-SALT model with rank* $n+2m$*, or a CP-SALT model with rank* $n+3m$*, can approximate the predictive mean of the steady-state LDS with error* $\|\hat{\mathbf{y}}_t^{(\mathsf{LDS})} - \hat{\mathbf{y}}_t^{(\mathsf{SALT})}\|_\infty = \mathcal{O}(\lambda_{\mathsf{max}}^L)$*.*

Proof. A stationary linear dynamical system (LDS) is defined as follows:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{b} + \boldsymbol{\epsilon}_t \tag{29}$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{d} + \boldsymbol{\delta}_t \tag{30}$$

where $\mathbf{y}_t \in \mathbb{R}^N$ is the t-th observation, $\mathbf{x}_t \in \mathbb{R}^D$ is the t-th hidden state, $\boldsymbol{\epsilon}_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\boldsymbol{\delta}_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{R})$, and $\boldsymbol{\theta} = (\mathbf{A}, \mathbf{b}, \mathbf{Q}, \mathbf{C}, \mathbf{d}, \mathbf{R})$ are the parameters of the LDS.

Following the notation of Murphy [19], the one-step-ahead posterior predictive distribution for the observations of the LDS defined above can be expressed as:

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d}, \mathbf{C}\boldsymbol{\Sigma}_{t|t-1}\mathbf{C}^T + \mathbf{R}) \tag{31}$$

where

$$\boldsymbol{\mu}_{t|t-1} = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{b} \tag{32}$$
$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t|t-1} + \mathbf{K}_t\mathbf{r}_t \tag{33}$$
$$\boldsymbol{\Sigma}_{t|t-1} = \mathbf{A}\boldsymbol{\Sigma}_{t-1}\mathbf{A}^T + \mathbf{Q} \tag{34}$$
$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\boldsymbol{\Sigma}_{t|t-1} \tag{35}$$
$$p(\mathbf{x}_1) = \mathcal{N}(\boldsymbol{\mu}_{1|0}, \boldsymbol{\Sigma}_{1|0}) \tag{36}$$
$$\mathbf{K}_t = (\boldsymbol{\Sigma}_{t|t-1}^{-1} + \mathbf{C}^T\mathbf{R}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}^{-1} \tag{37}$$
$$\mathbf{r}_t = \mathbf{y}_t - \mathbf{C}\boldsymbol{\mu}_{t|t-1} - \mathbf{d}. \tag{38}$$

We can then expand the mean $\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d}$ as follows:

$$\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d} = \mathbf{C}\sum_{l=1}^{t-1}\boldsymbol{\Gamma}_l\mathbf{A}\mathbf{K}_{t-l}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{t-1}\boldsymbol{\Gamma}_l(\mathbf{b} - \mathbf{A}\mathbf{K}_{t-l}\mathbf{d}) + \mathbf{d} \tag{39}$$

where

$$\boldsymbol{\Gamma}_l = \prod_{i=1}^{l-1}\mathbf{A}(\mathbf{I} - \mathbf{K}_{t-i}\mathbf{C}) \quad \text{for} \quad l \in \{2, 3, \ldots\}, \tag{40}$$
$$\boldsymbol{\Gamma}_1 = \mathbf{I}. \tag{41}$$

Theorem 3.3.3 of Davis and Vinter [22] (reproduced with our notation below) states that for a stabilizable and detectable system, the $\lim_{t\to\infty}\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is the unique solution of the discrete algebraic Riccati equation

$$\boldsymbol{\Sigma} = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T - \mathbf{A}\boldsymbol{\Sigma}\mathbf{C}^T(\mathbf{C}\boldsymbol{\Sigma}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\boldsymbol{\Sigma}\mathbf{A}^T + \mathbf{Q}. \tag{42}$$

As we are considering stable autonomous LDSs here, the system is stabilizable and detectable, as all unobservable states are themselves stable [22, 36]

Theorem 3.3.3 [Reproduced from Davis and Vinter [22], updated to our notation and context]**.** (a) If the pair $(\mathbf{A}, \mathbf{C})$ is detectable then there exists at least one non-negative solution to the discrete

algebraic Riccati equation (42).
(b) If further the pair $(\mathbf{A}, \mathbf{C})$ is stabilizable then this solution $\mathbf{\Sigma}$ is unique, and $\mathbf{\Sigma}_{t|t-1} \to \mathbf{\Sigma}$ as $t \to \infty$, where $\mathbf{\Sigma}_{t|t-1}$ is the sequence generated by (32)-(38) with arbitrary initial covariance $\mathbf{\Sigma}_0$. The matrix $\mathbf{\Gamma} = \mathbf{A}(\mathbf{I} - \mathbf{K}\mathbf{C})$ is stable, where $\mathbf{K}$ is the Kalman gain corresponding to $\mathbf{\Sigma}$, i.e.

$$\mathbf{K} = (\mathbf{\Sigma}^{-1} + \mathbf{C}^T \mathbf{R}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}^{-1} \tag{43}$$

Proof. See Davis and Vinter [22]. Note that Davis and Vinter [22] define the Kalman gain as $\mathbf{AK}$.

The convergence of the Kalman gain also implies that each term in the sequence $\mathbf{\Gamma}_l$ converges to

$$\mathbf{\Gamma}_l = \prod_{i=1}^{l-1} \mathbf{A}(\mathbf{I} - \mathbf{K}\mathbf{C}) = (\mathbf{A}(\mathbf{I} - \mathbf{K}\mathbf{C}))^{l-1} = \mathbf{\Gamma}^{l-1}, \tag{44}$$

where, concretely, we define $\mathbf{\Gamma} = \mathbf{A}(\mathbf{I} - \mathbf{K}\mathbf{C})$. We can therefore make the following substitution and approximation

$$\begin{aligned}
\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d} &\overset{\lim t\to\infty}{=} \mathbf{C}\sum_{l=1}^{t-1} \mathbf{\Gamma}^l \mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{t-1} \mathbf{\Gamma}^l (\mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}) + \mathbf{d} && (45)\\
&= \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l \mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l (\mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}) + \mathbf{d} + \sum_{l=L+1}^{\infty} \mathcal{F}\left(\mathbf{\Gamma}^l\right) && (46)\\
&\approx \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l \mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l (\mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}) + \mathbf{d} && (47)
\end{aligned}$$

The approximation is introduced as a result of truncating the sequence to consider just the “first” L terms, and discarding the higher-order terms (indicated in blue). It is important to note that each term in (45) is the sum of a geometric sequence multiplied elementwise with $\mathbf{y}_t$.

There are two components we prove from here. First, we derive an element-wise bound on the error introduced by the truncation, and verify that under the conditions outlined that the bound decays monotonically in L. We then show that Tucker and CP decompositions can represent the truncated summations in (47), and derive the minimum rank required for this representation to be exact.

Bounding The Error Term We first rearrange the truncated terms in (45), where we define $\mathbf{x}_l \triangleq \mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}$

$$\begin{aligned}
\sum_{l=L+1}^{\infty} \mathcal{F}\left(\mathbf{\Gamma}^l\right) &= \mathbf{C}\sum_{l=L+1}^{\infty} \mathbf{\Gamma}^l \mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=L+1}^{\infty} \mathbf{\Gamma}^l (\mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}) + \mathbf{d}, && (48)\\
&= \sum_{l=L+1}^{\infty} \mathbf{C}\mathbf{\Gamma}^l \mathbf{x}_l, && (49)\\
&= \sum_{l=L+1}^{\infty} \mathbf{C}\mathbf{E}\mathbf{\Lambda}^{l-1}\mathbf{E}^{-1}\mathbf{x}_l, && (50)\\
&= \sum_{l=L+1}^{\infty} \mathbf{P}\mathbf{\Lambda}^{l-1}\mathbf{q}_l, && (51)
\end{aligned}$$

where $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}$ is the eigendecomposition of $\mathbf{\Gamma}$, $\mathbf{P} \triangleq \mathbf{C}\mathbf{E}$, and $\mathbf{q}_l \triangleq \mathbf{E}^{-1}\mathbf{x}_l$. We now consider the infinity-norm of the error, and apply the triangle and Cauchy-Schwartz inequalities. We can write the bound on the as

$$\begin{aligned}
\epsilon &= \left| \left(\sum_{l=L+1}^{\infty} \mathcal{F}\left(\mathbf{\Gamma}^l\right) \right)_n \right|, \quad \text{where} \quad n = \arg\max_k \left| \left(\sum_{l=L+1}^{\infty} \mathcal{F}\left(\mathbf{\Gamma}^l\right) \right)_k \right| && (52)\\
&= \left| \sum_{l=L+1}^{\infty} \sum_{d=1}^{D} p_{nd}\lambda_d^{l-1} q_{l,d} \right|, && (53)\\
&\leq \sum_{l=L+1}^{\infty} \sum_{d=1}^{D} |p_{nd}| \left|\lambda_d^{l-1}\right| |q_{l,d}| \,. && (54)
\end{aligned}$$

Upper bounding the absolute magnitude of $q_{l,d}$ by W provides a further upper bound, which we can then rearrange

$$\epsilon \leq W \sum_{l=L+1}^{\infty} \sum_{d=1}^{D} |p_{nd}| \left|\lambda_d^{l-1}\right|, \tag{55}$$

$$= W \sum_{d=1}^{D} |p_{nd}| \sum_{l=L+1}^{\infty} \left|\lambda_d^{l-1}\right|. \tag{56}$$

The first two terms are constant, and hence the upper bound is determined by the sum of the of the l^{th} power of the eigenvalues. We can again bound this sum by setting all eigenvalues equal to the magnitude of the eigenvalue with the maximum magnitude (spectral norm), denoted λ_{max}:

$$\epsilon \leq W \sum_{d=1}^{D} |p_{nd}| \sum_{l=L+1}^{\infty} \lambda_{\max}^{l-1}, \tag{57}$$

where these second summation is not a function of d, and $W \sum_{d=1}^{D} |p_{nd}|$ is constant. This summation is a truncated geometric sequence. Invoking Theorem 3.3.3 of Davis and Vinter [22] again, the matrix $\mathbf{\Gamma}$ has only stable eigenvalues, and hence $\lambda_{\max} < 1$. Therefore the sequence sum will converge to

$$\sum_{l=L+1}^{\infty} \lambda_{\max}^{l-1} = \frac{\lambda_{\max}^{L}}{1-\lambda_{\max}}. \tag{58}$$

Rearranging again, we see that the absolute error on the n^{th} element of $\mathbf{y}_t$ is therefore bounded according to a power of the spectral norm

$$\epsilon \leq W \sum_{d=1}^{D} |p_{nd}| \frac{\lambda_{\max}^{L}}{1-\lambda_{\max}}, \tag{59}$$

$$= \mathcal{O}\left(\lambda_{\max}^{L}\right). \tag{60}$$

More specifically, for a stable linear time-invariant dynamical system, and where $\mathbf{q}$ — and hence $\mathbf{y}$ — is bounded, then the bound on the error incurred reduces exponentially in the length of the window L. Furthermore, this error bound will reduce faster for systems with a lower spectral norm.

Diagonalizing the System We first transform $\mathbf{\Gamma}$ into real modal form, defined as $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}$, where $\mathbf{E}$ and $\mathbf{\Lambda}$ are the eigenvectors and diagonal matrix of eigenvalues of $\mathbf{\Gamma}$. Letting $\mathbf{\Gamma}$ have n real eigenvalues and m pairs of complex eigenvalues (i.e., $n + 2m = D$), we can express $\mathbf{E}$, $\mathbf{\Lambda}$, and $\mathbf{E}^{-1}$ as:

$$\mathbf{E} = \left[\, \mathbf{a}_1 \,\ldots\, \mathbf{a}_n \;\mathbf{b}_1 \;\mathbf{c}_1 \,\ldots\, \mathbf{b}_m \;\mathbf{c}_m \,\right] \tag{61}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & & & & & & \\ & \ddots & & & & & & \\ & & \lambda_n & & & & & \\ & & & \sigma_1 & \omega_1 & & & \\ & & & -\omega_1 & \sigma_1 & & & \\ & & & & & \ddots & & \\ & & & & & & \sigma_m & \omega_m \\ & & & & & & -\omega_m & \sigma_m \end{bmatrix} \tag{62}$$

$$\mathbf{E}^{-1} = \begin{bmatrix} \mathbf{d}_1^T \\ \vdots \\ \mathbf{d}_n^T \\ \mathbf{e}_1^T \\ \boldsymbol{f}_1^T \\ \vdots \\ \mathbf{e}_m^T \\ \boldsymbol{f}_m^T \end{bmatrix} \tag{63}$$

where $\mathbf{a}_1 \dots \mathbf{a}_n$ are the right eigenvectors corresponding to n real eigenvalues $\lambda_1 \dots \lambda_n$, and $\mathbf{b}_i$ and $\mathbf{c}_i$ are the real and imaginary parts of the eigenvector corresponding to the complex eigenvalue $\sigma_i + j\omega_i$. Note that

$$\mathbf{\Gamma}^l = (\mathbf{A}(\mathbf{I} - \mathbf{KC}))^{l-1} = \mathbf{E}\mathbf{\Lambda}^{l-1}\mathbf{E}^{-1} \tag{64}$$

The l^{th} power of $\mathbf{\Lambda}$, $\mathbf{\Lambda}^l$, where $l \geq 0$, can be expressed as:

$$\mathbf{\Lambda}^l = \begin{bmatrix} \lambda_1^l & & & & & & & \\ & \ddots & & & & & & \\ & & \lambda_n^l & & & & & \\ & & & \sigma_{1,l} & \omega_{1,l} & & & \\ & & & -\omega_{1,l} & \sigma_{1,l} & & & \\ & & & & & \ddots & & \\ & & & & & & \sigma_{m,l} & \omega_{m,l} \\ & & & & & & -\omega_{m,l} & \sigma_{m,l} \end{bmatrix} \tag{65}$$

where $\sigma_{i,l} = \sigma_{i,l-1}^2 - \omega_{i,l-1}^2$, $\omega_{i,l} = 2\sigma_{i,l-1}\omega_{i,l-1}$ for $l \geq 2$, $\sigma_{i,1} = \sigma_i$, $\omega_{i,1} = \omega_i$, $\sigma_{i,0} = 1$, and $\omega_{i,0} = 0$.

Tucker Tensor Regression Let $\mathcal{H} \in \mathbb{R}^{D \times D \times L}$ be a three-way tensor, whose l^{th} frontal slice $\mathbf{H}_{::l} = \mathbf{\Lambda}^{l-1}$. Let $\mathcal{G} \in \mathbb{R}^{D \times D \times D}$ be a three-way tensor, whose entry $g_{ijk} = \mathbb{1}_{i=j=k}$ for $1 \leq k \leq n$, and $g_{ijk} = (-1)^{\mathbb{1}_{i+1=j=k+1}} \mathbb{1}_{(i=j=k)\vee(i-1=j-1=k)\vee(i=j+1=k+1)\vee(i+1=j=k+1)}$ for $k \in \{n+1, n+3, \dots, n+2m-1\}$. Let $\mathbf{W} \in \mathbb{R}^{L \times D}$ be a matrix, whose entry $w_{lk} = \lambda_k^{l-1}$ for $1 \leq k \leq n$, $w_{lk} = \sigma_{k,l-1}$ for $k \in \{n+1, n+3, \dots, n+2m-1\}$, and $w_{lk} = -\omega_{k,l-1}$ for $k \in \{n+2, n+4, \dots, n+2m\}$. We can then decompose $\mathcal{H}$ into $\mathcal{G} \in \mathbb{R}^{D \times D \times D}$ and $\mathbf{W} \in \mathbb{R}^{L \times D}$ such that $\mathcal{H} = \mathcal{G} \times_3 \mathbf{W}$ (Figure 6).

With $\mathbf{V} = (\mathbf{E}^{-1}\mathbf{AK})^T$, $\mathbf{U} = \mathbf{CE}$, $\mathbf{m} = \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l(\mathbf{b} - \mathbf{AKd}) + \mathbf{d}$, and $\mathbf{X}_t = \mathbf{y}_{t-1:t-L}$, we can rearrange the mean to:

$$\begin{aligned}
\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d} &\approx \mathbf{C}\sum_{l=1}^{L} \mathbf{E}\mathbf{\Lambda}^{l-1}\mathbf{E}^{-1}\mathbf{AK}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{L} \mathbf{\Gamma}^l(\mathbf{b} - \mathbf{AKd}) + \mathbf{d} && (66) \\
&= \mathbf{U}\sum_{l=1}^{L} \mathbf{H}_{::l}\mathbf{V}^T\mathbf{y}_{t-l} + \mathbf{m} && (67) \\
&= \mathbf{U}\sum_{l=1}^{L} (\mathcal{G}\bar{\times}_3\mathbf{w}_l)\mathbf{V}^T\mathbf{y}_{t-l} + \mathbf{m} && (68) \\
&= \mathbf{U}\sum_{l=1}^{L} ((\mathcal{G} \times_2 \mathbf{V})\bar{\times}_3\mathbf{w}_l)\mathbf{y}_{t-l} + \mathbf{m} && (69) \\
&= \mathbf{U}\sum_{l=1}^{L}\sum_{j=1}^{D}\sum_{k=1}^{D} \mathbf{g}_{:jk} \circ \mathbf{v}_{:j}(w_{lk}\mathbf{y}_{t-l}) + \mathbf{m} && (70) \\
&= \mathbf{U}\sum_{j=1}^{D}\sum_{k=1}^{D} \mathbf{g}_{:jk}(\mathbf{v}_{:j}^\top\mathbf{X}_t\mathbf{w}_{:k}) + \mathbf{m} && (71) \\
&= \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} \mathbf{u}_{:i}g_{ijk}(\mathbf{v}_{:j}^\top\mathbf{X}_t\mathbf{w}_{:k}) + \mathbf{m} && (72) \\
&= \left[\sum_{i=1}^{n+2m}\sum_{j=1}^{n+2m}\sum_{k=1}^{n+2m} g_{ijk}\mathbf{u}_{:i} \circ \mathbf{v}_{:j} \circ \mathbf{w}_{:k}\right] \times_{2,3} \mathbf{X}_t + \mathbf{m} && (73)
\end{aligned}$$

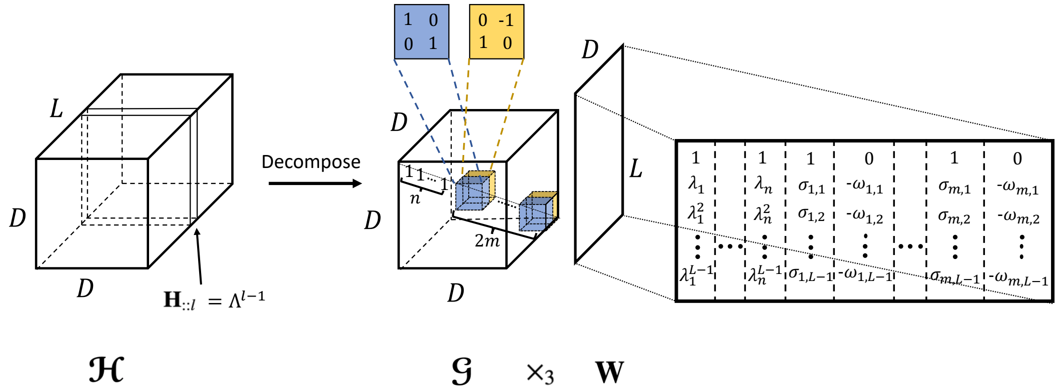


Figure 6: **Decomposition of $\mathcal{H}$ into $\mathcal{G}$ and $\mathbf{W}$ such that $\mathcal{H} = \mathcal{G} \times_3 \mathbf{W}$:** Given an LDS whose $\mathbf{A}(\mathbf{I}-\mathbf{KC})$ has n real eigenvalues and m pairs of complex eigenvalues, this decomposition illustrates how Tucker-SALT can approximate the LDS well with rank $n + 2m$.

CP Tensor Regression By rearranging $\mathbf{E}$, $\mathbf{\Lambda}^l$, and $\mathbf{E}^{-1}$ into $\mathbf{J}$, $\mathbf{P}_l$, and $\mathbf{S}$ respectively as follows:

$$\mathbf{J} = [\,\mathbf{a}_1 \;\dots\; \mathbf{a}_n \;\; \mathbf{b}_1+\mathbf{c}_1 \;\; \mathbf{b}_1 \;\; \mathbf{c}_1 \;\dots\; \mathbf{b}_m+\mathbf{c}_m \;\; \mathbf{b}_m \;\; \mathbf{c}_m\,] \tag{74}$$

$$\mathbf{P}_l = \begin{bmatrix} \lambda_1^l & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & \lambda_n^l & & & & & & & \\ & & & \sigma_{1,l} & & & & & & \\ & & & & \alpha_{1,l} & & & & & \\ & & & & & \beta_{1,l} & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & \sigma_{m,l} & & \\ & & & & & & & & \alpha_{m,l} & \\ & & & & & & & & & \beta_{m,l} \end{bmatrix} \tag{75}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{d}_1^T \\ \vdots \\ \mathbf{d}_n^T \\ \mathbf{e}_1^T + \boldsymbol{f}_1^T \\ \boldsymbol{f}_1^T \\ \boldsymbol{e}_1^T \\ \vdots \\ \mathbf{e}_m^T + \boldsymbol{f}_m^T \\ \boldsymbol{f}_m^T \\ \boldsymbol{e}_m^T \end{bmatrix} \tag{76}$$

where $\mathbf{J} \in \mathbb{R}^{D\times(n+3m)}$, $\mathbf{P}_l \in \mathbb{R}^{(n+3m)\times(n+3m)}$, $\mathbf{S} \in \mathbb{R}^{(n+3m)\times D}$, $\alpha_{i,l} = \omega_{i,l} - \sigma_{i,l}$, and $\beta_{i,l} = -\omega_{i,l} - \sigma_{i,l}$, we can diagonalize $(\mathbf{A}(\mathbf{I}-\mathbf{KC}))^l$ as $\mathbf{JP}_l\mathbf{S}$.

Let $\mathbf{V} = (\mathbf{SAK})^T$, $\mathbf{U} = \mathbf{CJ}$, $\mathbf{m} = \mathbf{C}\sum_{l=1}^{L}\mathbf{\Gamma}^l(\mathbf{b} - \mathbf{AKd}) + \mathbf{d}$, and $\mathbf{X}_t = \mathbf{y}_{t-1:t-L}$. Let $\mathbf{W} \in \mathbb{R}^{L\times(n+3m)}$ be a matrix, whose element in the l^{th} row and k^{th} column is $p_{l-1,kk}$ (i.e., the element in the k^{th} row and k^{th} column of $\mathbf{P}_{l-1}$), and $\mathcal{G} \in \mathbb{R}^{(n+3m)\times(n+3m)\times(n+3m)}$ be a superdiagonal 3-way tensor, where $g_{ijk} = \mathbb{1}_{i=j=k}$. We can then rearrange the mean to:

$$\begin{aligned}
\mathbf{C}\boldsymbol{\mu}_{t|t-1} + \mathbf{d} &\approx \mathbf{C}\sum_{l=1}^{L}\mathbf{E}\boldsymbol{\Lambda}^{l-1}\mathbf{E}^{-1}\mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{C}\sum_{l=1}^{L}\boldsymbol{\Gamma}^{l}(\mathbf{b} - \mathbf{A}\mathbf{K}\mathbf{d}) + \mathbf{d} && (77)\\
&= \mathbf{C}\sum_{l=1}^{L}\mathbf{J}\mathbf{P}_{l-1}\mathbf{S}\mathbf{A}\mathbf{K}\mathbf{y}_{t-l} + \mathbf{m} && (78)\\
&= \mathbf{U}\sum_{l=1}^{L}\mathbf{P}_{l-1}\mathbf{V}^{\top}\mathbf{y}_{t-l} + \mathbf{m} && (79)\\
&= \sum_{l=1}^{L}\sum_{i}^{n+3m}\sum_{j}^{n+3m}\sum_{k}^{n+3m} g_{ijk}\,\mathbf{u}_{:i}\circ\mathbf{v}_{:j}(p_{l-1,kk}\mathbf{y}_{t-l}) + \mathbf{m} && (80)\\
&= \sum_{i}^{n+3m}\sum_{j}^{n+3m}\sum_{k}^{n+3m} g_{ijk}\,\mathbf{u}_{:i}\circ\mathbf{v}_{:j}(\mathbf{X}_t\mathbf{w}_{:k}) + \mathbf{m} && (81)\\
&= \left[\sum_{i=1}^{n+3m}\sum_{j=1}^{n+3m}\sum_{k=1}^{n+3m} g_{ijk}\,\mathbf{u}_{:i}\circ\mathbf{v}_{:j}\circ\mathbf{w}_{:k}\right]\times_{2,3}\mathbf{X}_t + \mathbf{m} && (82)
\end{aligned}$$

And so concludes the proof. □

C Single-subspace SALT

Here we explicitly define the generative model of multi-subspace and single-subspace Tucker-SALT and CP-SALT. Single-subspace SALT is analogous to single-subspace SLDSs (also defined below), where certain emission parameters (e.g., $\mathbf{C}$, $\mathbf{d}$, and $\mathbf{R}$) are shared across discrete states. This reduces the expressivity of the model, but also reduces the number of parameters in the model. Note that both variants of all models have the same structure on the transition dynamics of $\mathbf{z}_t$.

Multi-subspace SALT Note that the SALT model defined in (6) and (7) in the main text was a multi-subspace SALT. We repeat the definition here for ease of comparison

$$\mathbf{y}_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(\left(\sum_{i=1}^{D_1}\sum_{j=1}^{D_2}\sum_{k=1}^{D_3} g_{ijk}^{(z_t)}\, \mathbf{u}_{:i}^{(z_t)} \circ \mathbf{v}_{:j}^{(z_t)} \circ \mathbf{w}_{:k}^{(z_t)}\right) \times_{2,3} \mathbf{y}_{t-1:t-L} + \mathbf{b}^{(z_t)}, \mathbf{\Sigma}^{(z_t)}\right), \tag{83}$$

$D_1 = D_2 = D_3 = D$ and $\mathcal{G}$ is diagonal for CP-SALT.

Single-subspace Tucker-SALT In single-subspace methods, the output factors are shared across discrete states

$$\mathbf{y}_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(\mathbf{U}\left(\mathbf{m}^{(z_t)} + \left(\sum_{j=1}^{D_2}\sum_{k=1}^{D_3} \mathbf{g}_{:jk}^{(z_t)} \circ \mathbf{v}_{:j}^{(z_t)} \circ \mathbf{w}_{:k}^{(z_t)}\right) \times_{2,3} \mathbf{y}_{t-1:t-L}\right) + \mathbf{b}, \mathbf{\Sigma}^{(z_t)}\right), \tag{84}$$

where $\mathbf{m}^{(z_t)} \in \mathbb{R}^{D_1}$.

Single-subspace CP-SALT Single-subspace CP-SALT requires an extra tensor compared to Tucker-SALT, as this tensor can no longer be absorbed in to the core tensor.

$$\mathbf{y}_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(\mathbf{U}'\left(\mathbf{m}^{(z_t)} + \mathbf{P}^{(z_t)}\left(\left(\sum_{j=1}^{D_2}\sum_{k=1}^{D_3} \mathbf{g}_{:jk}^{(z_t)} \circ \mathbf{v}_{:j}^{(z_t)} \circ \mathbf{w}_{:k}^{(z_t)}\right) \times_{2,3} \mathbf{y}_{t-1:t-L}\right)\right) + \mathbf{b}, \mathbf{\Sigma}^{(z_t)}\right), \tag{85}$$

where $\mathbf{U}' \in \mathbb{R}^{N \times D_1'}$, $\mathbf{P}^{(z_t)} \in \mathbb{R}^{D_1' \times D_1}$, $\mathbf{m}^{(z_t)} \in \mathbb{R}^{D_1'}$, $D_1 = D_2 = D_3 = D$, and $\mathcal{G}$ is diagonal.

Multi-subspace SLDS Multi-subspace SLDS is a much harder optimization problem, which we found was often numerically unstable. We therefore do not consider multi-subspace SLDS in these experiments, but include its definition here for completeness

$$\mathbf{x}_t \sim \mathcal{N}\left(\mathbf{A}^{(z_t)}\mathbf{x}_{t-1} + \mathbf{b}^{(z_t)},\, \mathbf{Q}^{(z_t)}\right), \tag{86}$$

$$\mathbf{y}_t \sim \mathcal{N}\left(\mathbf{C}^{(z_t)}\mathbf{x}_t + \mathbf{d}^{(z_t)},\, \mathbf{R}^{(z_t)}\right). \tag{87}$$

Single-subspace SLDS Single-subspace SLDS was used in all of our experiments, and is typically used in practice [8, 15]

$$\mathbf{x}_t \sim \mathcal{N}\left(\mathbf{A}^{(z_t)}\mathbf{x}_{t-1} + \mathbf{b}^{(z_t)},\, \mathbf{Q}^{(z_t)}\right), \tag{88}$$

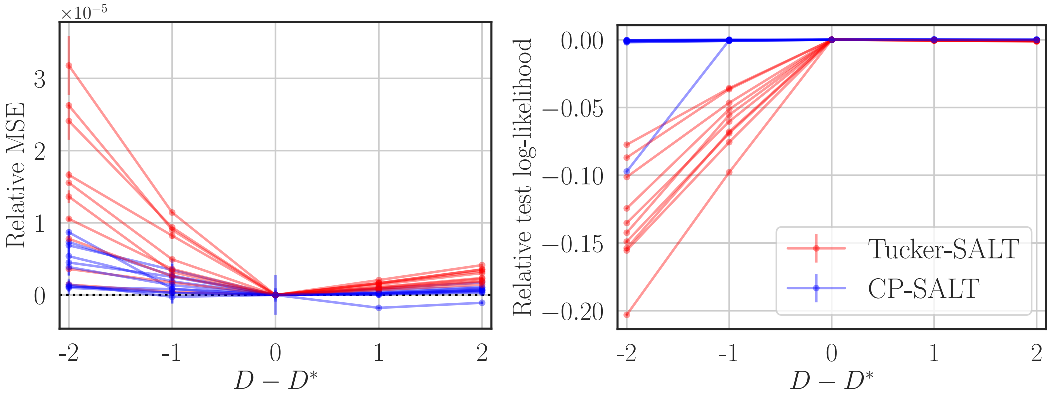
$$\mathbf{y}_t \sim \mathcal{N}\left(\mathbf{C}\mathbf{x}_t + \mathbf{d},\, \mathbf{R}\right). \tag{89}$$

D Synthetic Data Experiments

D.1 Extended Experiments for Proposition 1

In Section 5.1 we showed that Proposition 1 can accurately predict the required rank for CP- and Tucker-SALT models. We showed results for a single LDS for clarity. We now extend this analysis across multiple random LDS and SALT models. We randomly sampled LDSs with latent dimensions ranging from 4 to 10, and observation dimensions ranging from 9 to 20. For each LDS, we fit 5 randomly initialized CP-SALT and Tucker-SALT models with $L = 50$ lags. We varied the rank of our fit SALT models according to the rank predicted by Proposition 1. Specifically, we computed the estimated number of ranks for a given LDS, denoted D^*, and then fit SALT models with $\{D^* - 2, D^* - 1, D^*, D^* + 1, D^* + 2\}$ ranks. According to Proposition 1, we would expect to see the reconstruction error of the autoregressive tensor be minimized, and for prediction accuracy to saturate, at $D = D^*$.

To analyze these model fits, we first computed the average mean squared error of the autoregressive tensor corresponding to the LDS simulation, as a function of SALT rank relative to the rank required by Proposition 1. We see, as predicted by Proposition 1, that error in the autoregressive tensor is nearly always minimized at D^* (Figure 7A). Tucker-SALT was always minimized at D^*. Some CP-SALT fits have lower MSE at ranks other than predicted by Proposition 1. We believe this is due to local minima in the optimization. We next investigated the test log-likelihood as a function of the relative rank (Figure 7B). Interestingly, the test log-likelihood shows that Tucker-SALT strongly requires the correct number of ranks for accurate prediction, but CP-SALT can often perform well with fewer ranks than predicted (although still a comparable number of ranks to Tucker-SALT). As in Figure 2, these analyses empirically confirm Proposition 1.



(a) Normalized MSE of autoregressive tensor. *(b)* Normalized log-likelihood on held-out test set.

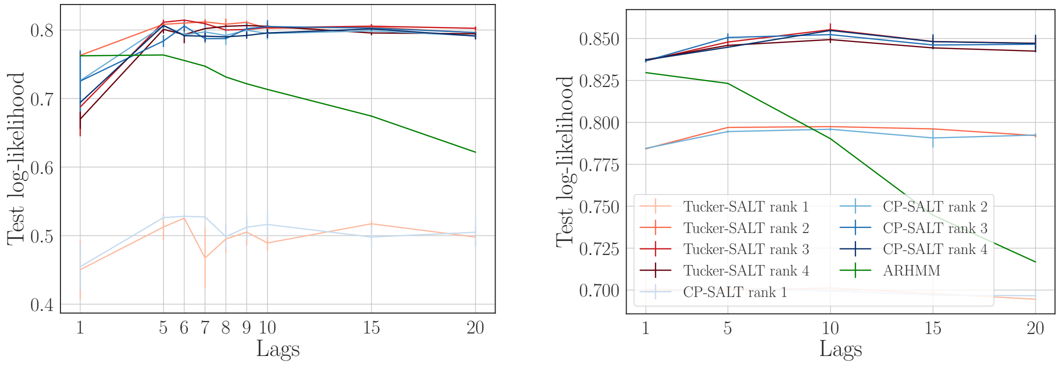
Figure 7: Extended results examining Proposition 1. Results are shown for the ability of SALT to estimate ten randomly generated LDSs, using five SALT repeats for each LDS. MSEs (in panel A) and log-likelihoods (in panel B) are normalized by the mean MSE and mean log-likelihood of SALT models trained with $D = D^*$. D is the rank of the fit SALT model, and D^* is the necessary rank predicted by Proposition 1.

D.2 Quantitative Performance: Synthetic Switching LDS Experiments

We include further results and analysis for the NASCAR® and Lorenz attractor experiments presented in Section 5.2. We compare the marginal likelihood achieved by single-subspace SALT models of different sizes. We see that SALT outperforms ARHMMs, and can fit larger models (more lags) without overfitting (Figure 8). Note that the SLDS does not admit exact inference, and so we cannot readily compute the exact marginal likelihood for the SLDS.

D.3 TVART versus SALT in recovering the parameters of SLDSs

We compared SALT to TVART [24], another tensor-based method for modeling autoregressive processes. We modified TVART (as briefly described in the original paper) so that it can handle AR(p) processes, as opposed to only AR(1) processes. TVART is also not a probabilistic model (i.e.,



(a) NASCAR.

(b) Lorenz.

Figure 8: Quantitative performance of different SALT models and ARHMMs (averaged over 3 different runs) on the synthetic experiments presented in Section 5.2. The test-set log likelihood is shown as a function of lags in the SALT model, for both **(A)** the NASCAR® and **(B)** Lorenz synthetic datasets.

cannot compute log-likelihoods), and so we focus our comparison on how well these methods recover the parameters of a ground-truth SLDS.

We used the same SLDS that we used to generate the NASCAR® dataset in Section 5.2. We then used $L = 7$ CP-SALT and Tucker-SALT with ranks 3 and 2, respectively, and computed the MSE between the ground truth tensor and SALT tensors. For TVART, we used $L = 7$, bin size of 10, and ranks 2 and 3 to fit the model to the data. We then clustered the inferred dynamics parameters to assign discrete states. To get the TVART parameter estimation, we computed the mean of the dynamics parameters for each discrete state and computed the MSE against the ground truth tensor. The MSE results are as follows:

Table 2: Results comparing SALT and TVART [24] on the NASCAR example.

Model	**Rank**	**Tensor Reconstruction MSE** ($\times 10^{-3}$)	**Number of parameters**
TVART	2	0.423	1.4K
TVART	3	0.488	2.0K
Tucker-SALT	2	0.294	0.6K
CP-SALT	3	0.297	0.7K

Table 2 shows that SALT models recover the dynamics parameters of the ground truth SLDS more accurately. Furthermore, we see that SALT models use fewer parameters than TVART models for the dataset (as the number of parameters in TVART scales linearly with the number of windows). We also note that TVART cannot be applied to held-out data, and, without post-hoc analysis, does not readily have a notion of re-usable dynamics or syllables.

D.4 The effect of the number of switches on the recovery of the parameters of the autoregressive dynamic tensors

We asked how the frequency of discrete state switches affected SALT's ability to recover the autoregressive tensors. We trained CP-SALT, Tucker-SALT, the ARHMM, all with $L = 5$ lags, and the SLDS on data sampled from an SLDS with varying number of discrete state switches. The ground-truth SLDS model had $H = 2$ discrete states, $N = 20$ observations and $D = 7$ dimensional continuous latent states. The matrix $\mathbf{A}^{(h)}(\mathbf{I} - \mathbf{K}^{(h)}\mathbf{C}^{(h)})$ of each discrete state of the ground-truth SLDS had 1 real eigenvalue and 3 pairs of complex eigenvalues. We sampled 5 batches of $T = 15,000$ timesteps of data from the ground-truth SLDS, with $s_n \in \{1, 10, 25, 75, 125\}$ number of discrete state switches that were evenly spaced out across the data. We then computed the mean squared error (MSE) between the SLDS tensors and the tensors reconstructed by SALT, the ARHMM, and the SLDS. (Figure 9). More precisely, we combined the 3rd order autoregressive tensors from each discrete state into a 4th order tensor, and calculated the MSE based on these 4th

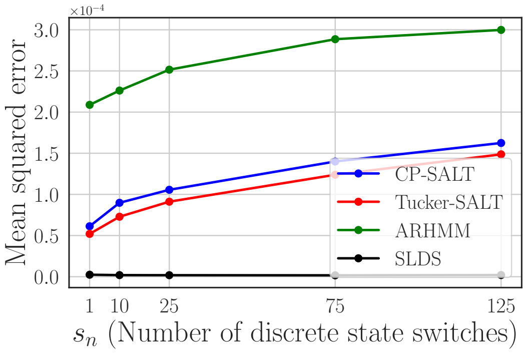


Figure 9: **The quality of SALT approximation of SLDSs decreases as the number of discrete state switches increases:** The data comes from an SLDS with $H = 2$, $N = 20$, and $D = 7$. 15,000 timesteps were generated, with varying numbers of evenly spaced discrete state switches (x-axis). The mean squared error of reconstructing the autoregressive tensors increased as a function of the number of discrete state switches. Note that we combined the 3rd order autoregressive tensors from each discrete state into a 4th order tensor, and calculated the MSE based on these 4th order tensors.

order tensors. As expected, the MSE increased with the number of switches in the data, indicating that the quality of SALT approximation of SLDSs decreases as the frequency of discrete state switches increases.

E Modeling Mouse Behavior

We include further details for the mouse experiments in Section 5.3.

E.1 Training Details

We used the first 35,949 timesteps of data from each of the three mice, which were collected at 30Hz resolution. We used $H = 50$ discrete states and fitted ARHMMs and CP-SALT models with varying lags and ranks. Similar to Wiltschko et al. [2], we imposed stickiness on the discrete state transition matrix via a Dirichlet prior with concentration of 1.1 on non-diagonals and 6×10^4 on the diagonals. These prior hyperparameters were empirically chosen such that the durations of the inferred discrete states and the given labels were comparable. We trained each model 5 times with random initialization for each hyperparameter, using 100 iterations of EM on a single NVIDIA Tesla P100 GPU.

E.2 Video Generation

Here we describe how the mouse behavioral videos were generated. We first determined the CP-SALT hyperparameters as those which led to the highest log-likelihood on the validation dataset. Then, using that CP-SALT model, we computed the most likely discrete states on the train and test data. Given a discrete state h, we extracted slices of the data whose most likely discrete state was h. We padded the data by 30 frames (i.e. 1 second) both at the beginning and the end of each slice for the movie. A red dot appears on each mouse for the duration of discrete state h. We generated such videos for all 50 discrete states (as long as there existed at least one slice for each discrete state) on the train and test data. For a given discrete state, the mice in each video behaved very similarly (e.g., the mice in the video for state 18 “pause" when the red dots appear, and those in the video for state 32 “walk" forward), suggesting that CP-SALT is capable of segmenting the data into useful behavioral syllables. See “MoSeq_salt_videos_train" and “MoSeq_salt_videos_test" in the supplementary material for the videos generated from the train and test data, respectively. “salt_crowd_i.mp4" refers to the crowd video for state i. We show the principal components for states $1, 2, 13, 32, 33, 47$ in Figure 10.

E.3 Modeling Mouse Behavior: Additional Analyses

We also investigated whether SALT qualitatively led to a good segmentation of the behavioral data into discrete states, shown in Figure 10. Figure 10A shows a 30 second example snippet of the test data from one mouse colored by the discrete states inferred by CP-SALT. CP-SALT used fewer discrete states to model the data than the ARHMM (Figure 10B). Coupled with the finding that CP-SALT improves test-set likelihoods, this suggests that the ARHMM may have oversegmented the data and CP-SALT may be better able to capture the number of behavioral syllables. Figure 10C shows average test data (with two standard deviations) for a short time window around the onset of a discrete state (we also include mouse videos corresponding to that state in the supplementary materials). The shrinking gray area around the time of state onset, along with the similar behaviors of the mice in the video, suggests that CP-SALT is capable of segmenting the data into consistent behavioral syllables.

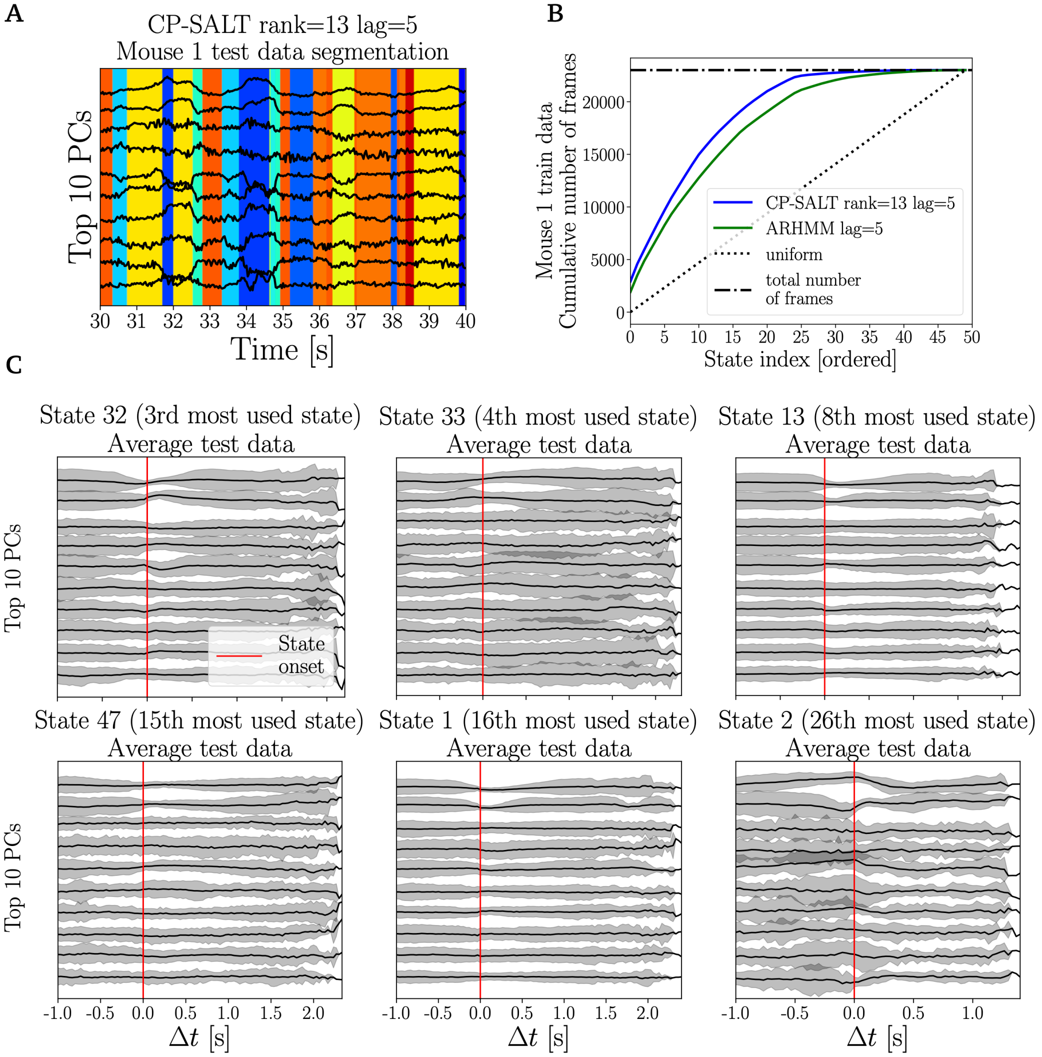


Figure 10: **CP-SALT leads to qualitatively good segmentation of the mouse behavioral data into distinct syllables.**: **(A)** 30 seconds of test data (Mouse 1) with the discrete states inferred by CP-SALT as the background color. **(B)** For one mouse, the cumulative number of frames that are captured by each discrete state, where the discrete states are ordered according to how frequently they occur. **(C)** The average test data, with two standard deviations, for six states of CP-SALT, aligned to the time of state onset. The shrinkage of the gray region around the state onset tells us that CP-SALT segments the test data consistently.

F Modeling *C. elegans* Neural Data

We include further details and results for the *C. elegans* example presented in Section 5.4. This example highlights how SALT can be used to gain scientific insight in to the system.

F.1 Training Details

We used ∼3200 timesteps of data (recorded at 3Hz) from one worm, for which 48 neurons were confidently identified. The data were manually segmented in to seven labels (reverse sustained, slow, forward, ventral turn, dorsal turn, reversal (type 1) and reversal (type 2). We therefore used $H = 7$ discrete states in all models (apart from the GLM). After testing multiple lag values, we selected $L = 9$ for all models, as these longer lags allow us to examine longer-timescale interactions and produced better segmentations across models, with only a small reduction in variance explained. We trained each model 5 times with KMeans initialization, using 100 iterations of EM on a single NVIDIA Tesla V100 GPU. Models that achieved 90% explained variance on a held-out test set were then selected and analyzed (similar to Linderman et al. [9]).

F.2 Additional Quantitative Results

Figure 11 shows additional results for training different models. In Figure 11A we see that models with larger ranks (or latent dimension) achieve higher explained variance. Interestingly, longer lags can lead to a slight reduction in the explained variance, likely due to overfitting. This effect is less pronounced in the more constrained single-subspace SALT, but, these models achieve lower explained variance ratios throughout. Longer lag models allow us to inspect longer-timescale dependencies, and so are more experimentally insightful. Figure 11B shows the confusion matrix for discrete states between learned models and the given labels. The segmentations were similar across all models that achieved 90% explained variance.

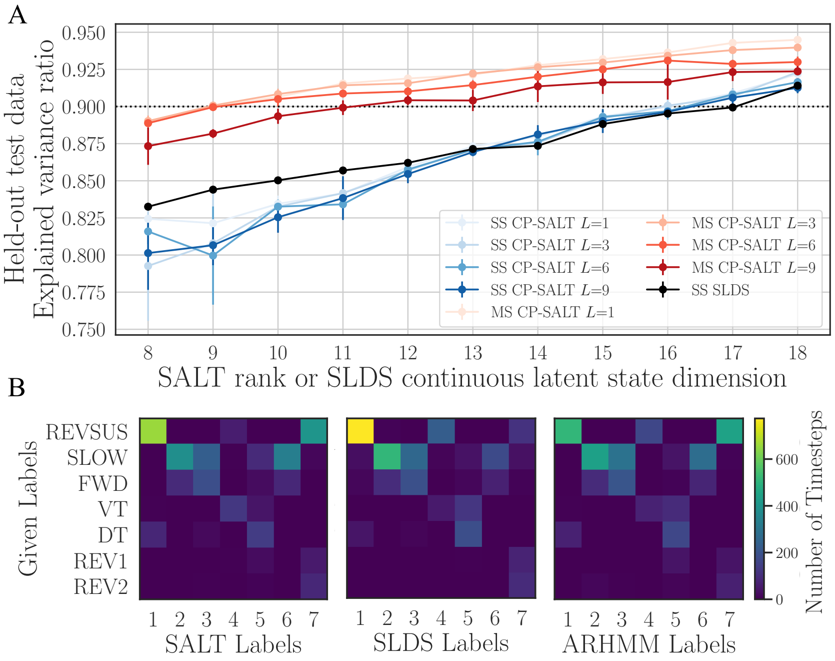


Figure 11: **SALT and SLDS perform comparably on held-out data**: **(A)**: Explained variance on a held-out sequence. Single-subspace (SS) SALT and SLDS perform comparably. Multi-subspace (MS) SALT achieves a higher explained variance with fewer ranks. Multi-subspace SLDS was numerically unstable. **(B)**: Confusion matrices between given labels and predicted labels. All methods produce similar quality segmentations.

F.3 Additional Autoregressive Filters

Figures 12 and 13 show extended versions of the autoregressive filters included in Section 5.4. Figure 12 shows the filters learned for ventral and dorsal turns (for which panel A was included in Figure 5), while Figure 13 shows the filters for forward and backward locomotion. Note that the GLM does not have multiple discrete states, and hence the same filters are used across states. We see for ARHMM and SALT that known-behavior tuned neurons have higher magnitude filters (determined by area under curve), whereas the SLDS and GLM do not recover such strong state-specific tuning. Since the learned SLDS did not have stable within-state dynamics, the autoregressive filters could not be computed using Equation (47). We thus show $CA^{(h)l}C^{+}$ for lag l as a proxy for the autoregressive filters of discrete state h of the SLDS. Note that this is a post-hoc method and does not capture the true dependencies in the observation space.

We see that SALT consistently assigns high autoregressive weight to neurons known to be involved in certain behaviors (see Figures 12 and 13). In contrast, the ARHMM identifies these relationships less reliably, and the estimate of the SLDS autoregressive filters identifies few strong relationships. As the GLM only have one "state", the autoregressive filters are averaged across state, and so few strong relationships are found. This highlights how the low-rank and switching properties of SALT can be leveraged to glean insight into the system.

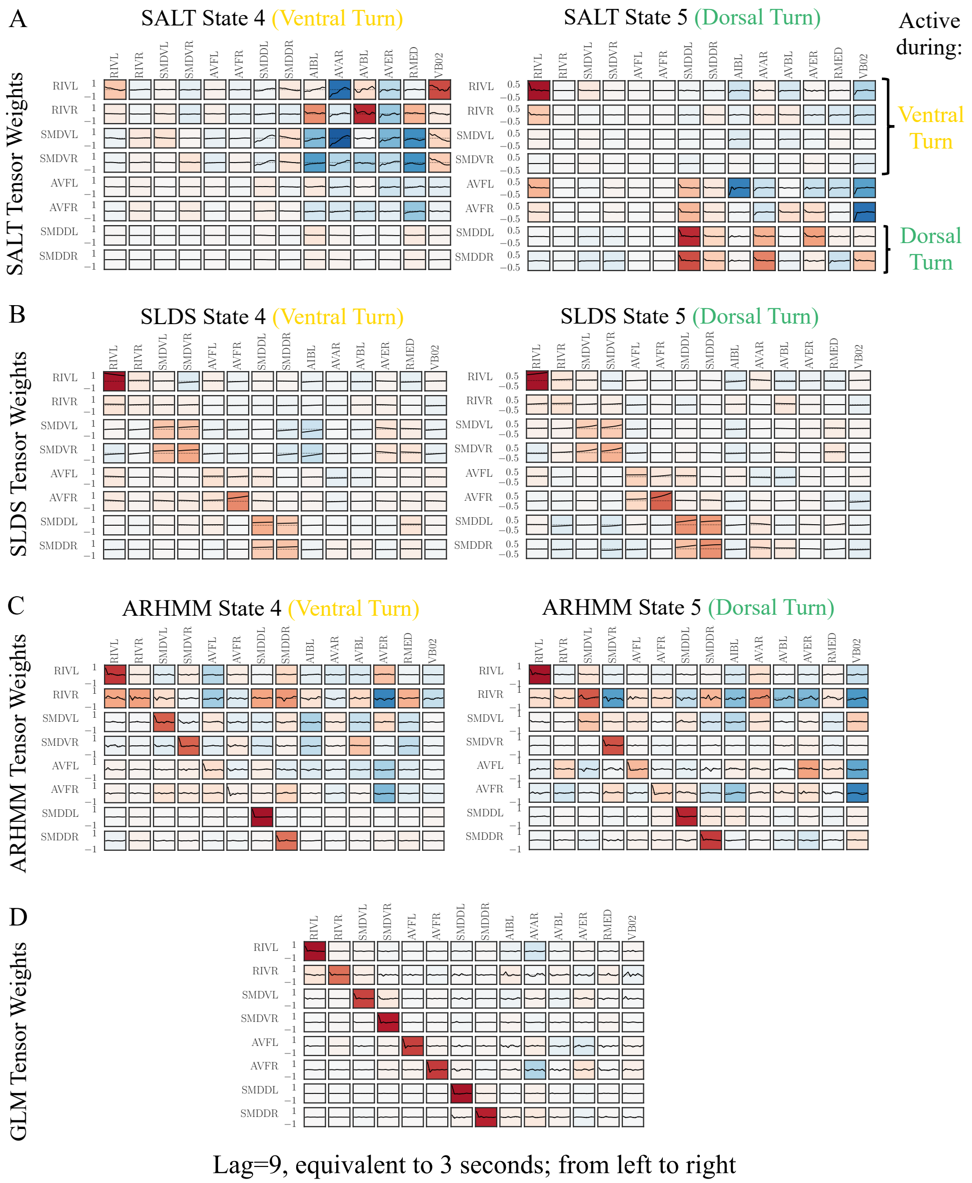


Figure 12: **Autoregressive tensors learned by different models (Ventral and Dorsal Turns)**: **(A-C)** One-dimensional autoregressive filters learned in two states by SALT, SLDS, ARHMM (identified as ventral and dorsal turns), and **(D)** by a GLM. `RIV` and `SMDV` are known to mediate ventral turns, while `SMDD` mediate dorsal turns [31, 32, 34].

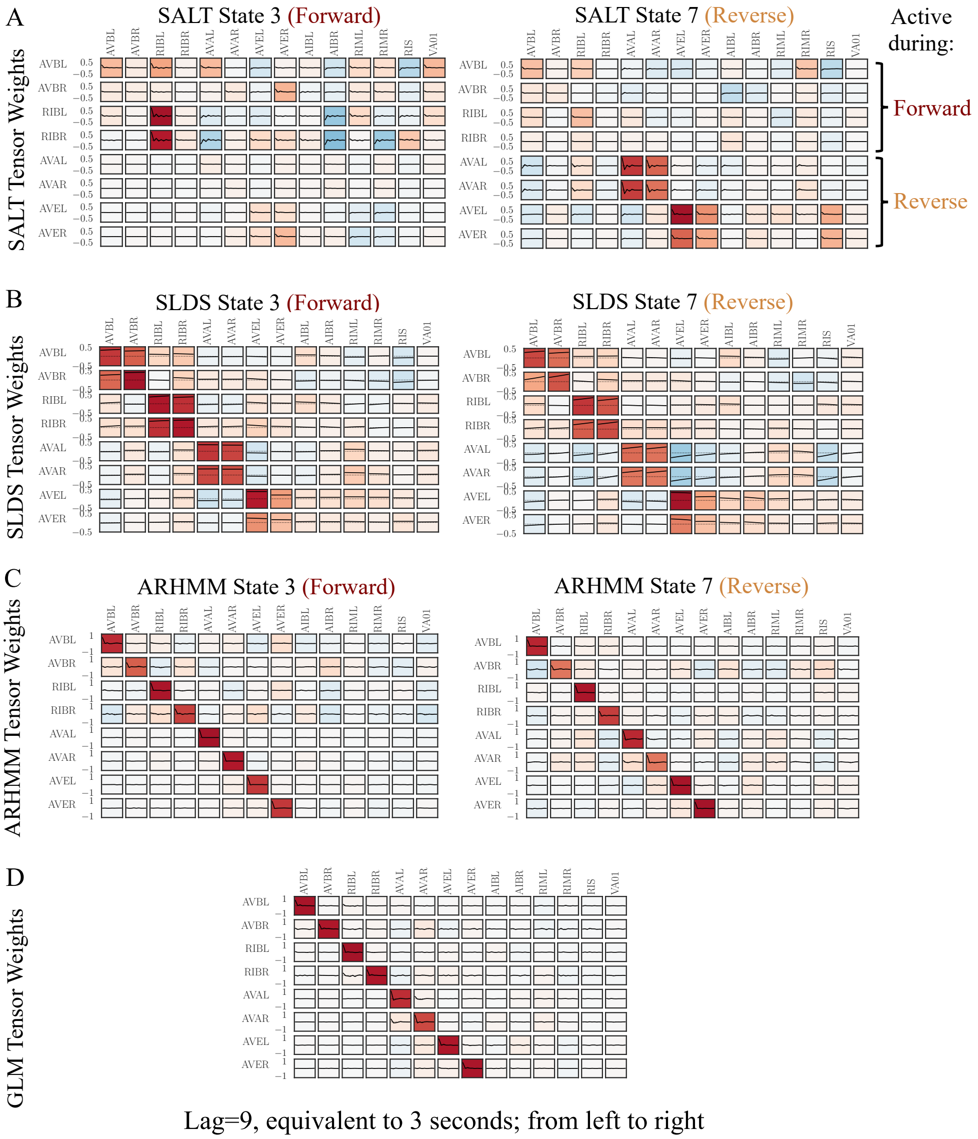


Figure 13: **Autoregressive tensors learned by different models (Forward Locomotion and Reversal)**: **(A-C)** One-dimensional autoregressive filters learned in two states by SALT, SLDS, ARHMM (identified as forward and reverse), and **(D)** by a GLM. `AVB` and `RIB` are known to mediate forward locomotion, while `AVA` and `AVE` are involved in initiating reversals [31, 32, 37, 38].

G Code Availability

As part of Supplementary materials, we include the source code for SALT (in Python) and an example Jupyter Notebook. This notebook shows how to sampling data from SALT, and then re-fit a SALT model to the data with an EM algorithm. We include just the core code here for ease of inspection, and will release the full source code after the review process.