# Technical Description of Dirichlet Tucker Decomposition

Libby Zhang and Scott Linderman

## 1 Model

Let $\mathcal{X} \in \mathbb{N}^{M \times N \times P \times S}$ denote a four-dimensional tensor of non-negative counts $x_{m,n,p,s}$ for each mouse $m = 1, \ldots, M$, epoch $n = 1, \ldots, N$, position bin $p = 1, \ldots, P$, and behavioral syllable $s = 1, \ldots, S$. Since there are a fixed number of frames of video, $C$, for each mouse $m$ and epoch $n$ (i.e., since all epochs are the same length), the faces of the tensor, $\boldsymbol{X}_{m,n} = [[x_{m,n,p,s}]] \in \mathbb{N}^{P \times S}$ have a fixed sum,

$$\sum_{p=1}^{P} \sum_{s=1}^{S} x_{m,n,p,s} = C \quad \forall m, n.$$

We propose a non-negative tensor decomposition that respects this constraint.

First, define the following model parameters. Let,

- $\boldsymbol{\psi}_m \in \Delta_{K_M}$ denote the $m$-th **mouse loading**,
- $\boldsymbol{\phi}_n \in \Delta_{K_N}$ denote the $n$-th **epoch loading**,
- $\boldsymbol{\theta}_k \in \Delta_P$ for $k = 1, \ldots, K_P$ denote the $k$-th **position factor**,
- $\boldsymbol{\lambda}_\ell \in \Delta_S$ for $\ell = 1, \ldots, K_S$ denote the $\ell$-th **syllable factor**, and
- $\mathcal{G} \in \mathbb{R}_+^{K_M \times K_N \times K_P \times K_S}$ denote the **core tensor** with entries $g_{i,j,k,\ell}$ and faces $\boldsymbol{G}_{i,j} = [[g_{i,j,k,\ell}]] \in \mathbb{R}_+^{K_P \times K_S}$.

In our model, the faces of the core tensor must be normalized such that,

$$\sum_{k=1}^{K_P} \sum_{\ell=1}^{K_S} g_{i,j,k,\ell} = 1 \tag{1}$$

for all $i = 1 \ldots, K_M$ and $j = 1, \ldots, K_N$.

We model the data as realizations of a multinomial distribution,

$$\mathrm{vec}(\boldsymbol{X}_{m,n}) \sim \mathrm{Mult}\left( C, \sum_{i=1}^{K_M} \sum_{j=1}^{K_N} \sum_{k=1}^{K_P} \sum_{\ell=1}^{K_S} g_{i,j,k,\ell} \psi_{m,i} \phi_{n,j} \mathrm{vec}(\boldsymbol{\theta}_k \boldsymbol{\lambda}_\ell^\top) \right). \tag{2}$$

To check that the multinomial parameter is properly normalized, note that,

$$\begin{aligned}
\sum_{p=1}^{P}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s} &= \sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\left(\sum_{p=1}^{P}\theta_{k,p}\right)\left(\sum_{s=1}^{S}\lambda_{\ell,s}\right) \\
&= \sum_{i=1}^{K_M}\psi_{m,i}\sum_{j=1}^{K_N}\phi_{n,j}\left(\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\right) \\
&= \sum_{i=1}^{K_M}\psi_{m,i}\sum_{j=1}^{K_N}\phi_{n,j} \\
&= 1,
\end{aligned}$$

where the third line follows from the assumption in eq. (1).

**Prior Distributions:** We place Dirichlet priors on the parameters,[1]

$$\begin{aligned}
\boldsymbol{\psi}_m &\overset{\text{iid}}{\sim} \mathrm{Dir}(\alpha_\psi \mathbf{1}_{K_M}) \\
\boldsymbol{\phi}_n &\overset{\text{iid}}{\sim} \mathrm{Dir}(\alpha_\phi \mathbf{1}_{K_N}) \\
\boldsymbol{\theta}_k &\overset{\text{iid}}{\sim} \mathrm{Dir}(\alpha_\theta \mathbf{1}_{P}) \\
\boldsymbol{\lambda}_\ell &\overset{\text{iid}}{\sim} \mathrm{Dir}(\alpha_\theta \mathbf{1}_{S}) \\
\mathrm{vec}(\boldsymbol{G}_{i,j}) &\overset{\text{iid}}{\sim} \mathrm{Dir}(\alpha_g \mathbf{1}_{K_P \cdot K_S}).
\end{aligned}$$

The EM algorithm described below requires $\alpha > 1$. In practice, we set $\alpha_\star = 1.1$ for all parameters.

### 1.1 Connection to Tucker Decompositions

Under this model,

$$\mathbb{E}[x_{m,n,p,s}] = C\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}$$

More compactly,

$$\mathbb{E}[\mathscr{X}] = C \cdot \mathscr{G} \circ \boldsymbol{\Psi} \circ \boldsymbol{\Phi} \circ \boldsymbol{\Theta} \circ \boldsymbol{\Lambda},$$

where $\boldsymbol{\Psi} \in \mathbb{R}_+^{M\times K_M}$ is a matrix with *rows* $\boldsymbol{\psi}_m$, $\boldsymbol{\Phi} \in \mathbb{R}_+^{N\times K_N}$ is a matrix with *rows* $\boldsymbol{\phi}_n$, $\boldsymbol{\Theta} \in \mathbb{R}_+^{P\times K_P}$ is a matrix with *columns* $\boldsymbol{\theta}_k$, $\boldsymbol{\Lambda} \in \mathbb{R}_+^{S\times K_S}$ is a matrix with *columns* $\boldsymbol{\lambda}_\ell$, and $\circ$ denotes a tensor-matrix multiplication. We recognize this as a 4-dimensional Tucker decomposition [Kolda and Bader, 2009] with non-negativity and normalization constraints on the factors enforced by Dirichlet priors. Hence, we call this model a **Dirichlet Tucker Decomposition**.

[1] $\Delta_K$ denotes the $(K-1)$-dimensional probability simplex embedded in $\mathbb{R}^K$.

### 1.2 Data Augmentation

To facilitate parameter inference, we augment the model by leveraging the Poisson/multinomial relationship.

$$
\begin{aligned}
x_{m,n,p,s} &\sim \mathrm{Po}\left(\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}\right) \\
&\Rightarrow \mathrm{vec}(\boldsymbol{X}_{m,n}) \mid (\mathbf{1}^\top \boldsymbol{X}_{m,n}\mathbf{1} = C) \sim \mathrm{Mult}\left(C, \sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\mathrm{vec}(\boldsymbol{\theta}_k\boldsymbol{\lambda}_\ell^\top)\right)
\end{aligned}
$$

We can "thin" the Poisson counts into those arising from each of the terms in the sum,

$$
\begin{aligned}
z_{m,n,p,s,i,j,k,\ell} &\overset{\text{ind}}{\sim} \mathrm{Po}(g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}) \\
&\Rightarrow x_{m,n,p,s} = \left(\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}\right) \sim \mathrm{Po}\left(\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}\right)
\end{aligned}
$$

These relationships permit us to think of the multinomial observation model as arising from a sum of independent Poisson counts, conditioned on the total count summing to $C$.

We augment the model with a tensor of latent counts, $\mathscr{Z} \in \mathbb{N}^{M\times N\times P\times S\times K_M\times K_N\times K_P\times K_S}$, with entries $z_{m,n,p,s,i,j,k,\ell}$ defined above. In the augmented model, the complete data log likelihood is,

$$
\begin{aligned}
p(\mathscr{X}, \mathscr{Z} \mid \mathscr{G}, \boldsymbol{\Psi}, \boldsymbol{\Phi}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}) = &\prod_{m=1}^{M}\prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S} \mathbb{I}\left[x_{m,n,p,s} = \sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}\right] \\
&\prod_{m=1}^{M}\prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S}\prod_{i=1}^{K_M}\prod_{j=1}^{K_N}\prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} \mathrm{Po}(z_{m,n,p,s,i,j,k,\ell} \mid g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s})
\end{aligned}
$$

The key idea is that in the augmented model, the parameters can be straightforwardly estimated via expectation-maximization (EM).

## 2 *Maximum a posteriori* (MAP) estimation

To estimate the model parameters, we maximize the probability in eq. (2) using the expectation-maximization (EM) algorithm. The algorithm alternates between the E- and M-steps described below.

### 2.1 E-step

The E-step is to compute the posterior distribution of the latent variables $\mathscr{Z}$,

$$
p(\mathscr{Z} \mid \mathscr{G}, \boldsymbol{\Psi}, \boldsymbol{\Phi}, \boldsymbol{\Theta}, \mathscr{X}) = \prod_{m=1}^{M}\prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S} \mathrm{Mult}(\boldsymbol{z}_{m,n,p,s} \mid x_{m,n,p,s}, \boldsymbol{\pi}_{m,n,p,s})
$$

where $\boldsymbol{\pi}_{m,n,p,s} \in \Delta^{K_M \cdot K_N \cdot K_P \cdot K_S}$ has entries,

$$\pi_{m,n,p,s,i,j,k,\ell} = \frac{g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}}{\sum_{i'=1}^{K_M}\sum_{j'=1}^{K_N}\sum_{k'=1}^{K_P}\sum_{\ell'=1}^{K_S} g_{i',j',k',\ell'}\psi_{m,i'}\phi_{n,j'}\theta_{k',p}\lambda_{\ell',s}}$$

For the M-steps below, we only need to compute the expected value of these augmentation variables,

$$\mathbb{E}[z_{m,n,p,s,i,j,k,\ell}] = x_{m,n,p,s}\pi_{m,n,p,s,i,j,k,\ell}.$$

## 2.2 M-step

In the M-step, we maximize the expected log joint probability under the posterior over $\mathscr{Z}$. We will do so via coordinate ascent, iteratively maximizing the expected log conditional distribution for each parameter, one at a time.

**M-step for $\boldsymbol{\psi}_m$** Fixing the other parameters,

$$\begin{aligned}
p(\boldsymbol{\psi}_m \mid -) &\propto \mathrm{Dir}(\boldsymbol{\psi}_m \mid \alpha_\psi \mathbf{1}_{K_M}) \prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S}\prod_{i=1}^{K_M}\prod_{j=1}^{K_N}\prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} \mathrm{Po}(z_{m,n,p,s,i,j,k,\ell} \mid g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}) \\
&\propto \prod_{i=1}^{K_M} \psi_{m,i}^{\alpha_\psi - 1} \prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S}\prod_{i=1}^{K_M}\prod_{j=1}^{K_N}\prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} \psi_{m,i}^{z_{m,n,p,s,i,j,k,\ell}} e^{-g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}}\, \mathbb{I}[\boldsymbol{\psi}_m \in \Delta_{K_M}] \\
&\propto \prod_{i=1}^{K_M} \psi_{m,i}^{\alpha_{m,i}-1} \mathbb{I}[\boldsymbol{\psi}_m \in \Delta_{K_M}] \\
&= \mathrm{Dir}(\boldsymbol{\psi}_m \mid \boldsymbol{\alpha}_m)
\end{aligned}$$

where

$$\alpha_{m,i} = \alpha_\psi + \sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{s=1}^{S}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}$$

and $\boldsymbol{\alpha}_m = (\alpha_{m,1}, \ldots, \alpha_{m,K_M})^\top$. The simplification arises thanks to the normalization constraints on the data and parameters.

The M-step maximizes the expected log probability under the posterior distribution of the augmentation variables. The maximum is at the mode of a Dirichlet distribution with parameters $\mathbb{E}[\boldsymbol{\alpha}_m]$,

$$\psi_{m,i}^\star = \frac{\mathbb{E}[\alpha_{m,i} - 1]}{\sum_{i'=1}^{K_M}\mathbb{E}[\alpha_{m,i'} - 1]}$$

Since $\alpha_\psi > 1$, the mode is guaranteed to exist.

The updates for the other parameters follow by symmetry.

**M-step for $\boldsymbol{\phi}_n$** By symmetry,

$$p(\boldsymbol{\phi}_n \mid -) \propto \mathrm{Dir}(\boldsymbol{\phi}_n \mid \boldsymbol{\alpha}_n)$$

where $\boldsymbol{\alpha}_n = (\alpha_{n,1}, \dots, \alpha_{n,K_N})^\top$ with,

$$\alpha_{n,j} = \alpha_\phi + \sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{k=1}^{K_P}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}.$$

**M-step for $\boldsymbol{\theta}_k$** By symmetry,

$$p(\boldsymbol{\theta}_k \mid -) \propto \mathrm{Dir}(\boldsymbol{\theta}_k \mid \boldsymbol{\alpha}_k)$$

where $\boldsymbol{\alpha}_k = (\alpha_{k,1}, \dots, \alpha_{k,P})^\top$ with,

$$\alpha_{k,p} = \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}.$$

**M-step for $\boldsymbol{\lambda}_\ell$** By symmetry,

$$p(\boldsymbol{\lambda}_\ell \mid -) \propto \mathrm{Dir}(\boldsymbol{\lambda}_\ell \mid \boldsymbol{\alpha}_\ell)$$

where $\boldsymbol{\alpha}_\ell = (\alpha_{\ell,1}, \dots, \alpha_{\ell,S})^\top$ with,

$$\alpha_{\ell,s} = \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{k=1}^{K_P} z_{m,n,p,s,i,j,k,\ell}.$$

**M-step for $\boldsymbol{G}_{i,j}$** It is less obviously symmetric, but the updates for the faces of the core tensor follow the same form.

$$\begin{aligned}
p(\boldsymbol{G}_{i,j} \mid -) &\propto \mathrm{Dir}(\mathrm{vec}(\boldsymbol{G}_{i,j}) \mid \alpha_g \mathbf{1}_{K_P \cdot K_S}) \prod_{m=1}^{M}\prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S}\prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} \mathrm{Po}(z_{m,n,p,s,i,j,k,\ell} \mid g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}) \\
&\propto \prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} g_{i,j,k,\ell}^{\alpha_g - 1} \prod_{m=1}^{M}\prod_{n=1}^{N}\prod_{p=1}^{P}\prod_{s=1}^{S}\prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} g_{i,j,k,\ell}^{z_{m,n,p,s,i,j,k,\ell}} e^{-g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}} \,\mathbb{I}[\mathrm{vec}(\boldsymbol{G}_{i,j}) \in \Delta_{K_P \cdot K_S}] \\
&\propto \prod_{k=1}^{K_P}\prod_{\ell=1}^{K_S} g_{i,j,k,\ell}^{\alpha_{i,j,k,\ell} - 1} \,\mathbb{I}[\mathrm{vec}(\boldsymbol{G}_{i,j}) \in \Delta_{K_P \cdot K_S}] \\
&= \mathrm{Dir}(\mathrm{vec}(\boldsymbol{G}_{i,j}) \mid \mathrm{vec}(\boldsymbol{A}_{i,j}))
\end{aligned}$$

where $\boldsymbol{A}_{i,j} = [[\alpha_{i,j,k,\ell}]] \in \mathbb{R}_+^{K_P \times K_S}$ with,

$$\alpha_{i,j,k,\ell} = \alpha_g + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{s=1}^{S} z_{m,n,p,s,i,j,k,\ell}.$$

### 2.3 EM with collapsed allocation tensor

The augmented tensor of expected latent counts, $\mathscr{Z} \in \mathbb{N}^{M\times N\times P\times S\times K_M\times K_N\times K_P\times K_S}$, greatly simplifies the estimation of the model parameters, but it incurs a significant memory footprint. We note this counts allocation tensor is immediately collapsed during the M-step of each parameter, suggesting that we do not necessarily need to instantiate the tensor in its entirety during implementation.

For example, consider the M-step for $\boldsymbol{\theta}_k$. In the derivation above, we found that expected log conditional probability is maximized at the mode of the Dirichlet distribution with parameter $\mathbb{E}[\boldsymbol{\alpha}_k]$ for pseudo-counts vector $\boldsymbol{\alpha}_k \in \mathbb{R}_+^P$ with elements

$$\alpha_{k,p} = \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}.$$

The expected value is,

$$\begin{aligned}
\mathbb{E}\left[\alpha_{k,p}\right] &= \alpha_\theta + \mathbb{E}\left[\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} z_{m,n,p,s,i,j,k,\ell}\right] \\
&= \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} \mathbb{E}[z_{m,n,p,s,i,j,k,\ell}] \\
&= \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} \left(x_{m,n,p,s}\pi_{m,n,p,s,i,j,k,\ell}\right) \\
&= \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\left(x_{m,n,p,s}\left(\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S}\pi_{m,n,p,s,i,j,k,\ell}\right)\right).
\end{aligned}$$

Expanding the definition of $\pi_{m,n,p,s,i,j,k,\ell}$, the inner sum simplifies to

$$\sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S}\pi_{m,n,p,s,i,j,k,\ell} = \frac{r_{m,n,p,s,k}}{C_{m,n,p,s}}$$

where

$$C_{m,n,p,s} = \sum_{i'=1}^{K_M}\sum_{j'=1}^{K_N}\sum_{k'=1}^{K_P}\sum_{\ell'=1}^{K_S} g_{i',j',k',\ell'}\psi_{m,i'}\phi_{n,j'}\theta_{k',p}\lambda_{\ell',s}$$

is the normalizing constant, and

$$r_{m,n,p,s,k} = \sum_{i=1}^{K_M}\sum_{j=1}^{K_N}\sum_{\ell=1}^{K_S} g_{i,j,k,\ell}\psi_{m,i}\phi_{n,j}\theta_{k,p}\lambda_{\ell,s}$$

is the collapsed allocation tensor. Given these two quantities, we can compute the expectation as,

$$\mathbb{E}\left[\alpha_{k,p}\right] = \alpha_\theta + \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{s=1}^{S}\left(\frac{x_{m,n,p,s}\, r_{m,n,p,s,k}}{C_{m,n,p,s}}\right).$$

Note that the temporary variables required for this computation use only $\mathcal{O}(MNPS)$ and $\mathcal{O}(MNPSK_P)$ memory, respectively — a dramatic reduction from the memory required to store $\mathscr{Z}$ naively.

## 3 Model Selection

The main hyperparameters to be determined are the number of factors, $K_M$, $K_N$, $K_P$, and $K_S$. We choose these parameters using cross-validation using a random, speckled test set. Specifically, we hold out a random subset of faces $\boldsymbol{X}_{m,n}$ from the data; i.e., we mask a random subset of (mouse, epoch) pairs. That way, we still have enough observed data to estimate the mouse loadings, $\boldsymbol{\psi}_m$, for each mouse, and the epoch loadings, $\boldsymbol{\phi}_n$, for each epoch. In the algorithms above, we can incorporate the mask by fixing the augmentation varaiables $z_{m,n,p,s,i,j,k,\ell}$ to zero whenever the index $(m, n)$ is held out. We evaluate the log likelihood of the held out data under the multinomial model in eq. (2), using the estimated parameters.

We sweep over a four-dimensional grid of numbers of factors $K_M \in \{2, 4, \ldots, 24\}$, $K_N \in \{2, 4, \ldots, 8\}$, $K_P \in \{2, 4, \ldots, 8\}$, and $K_S \in \{2, 4, \ldots, 24\}$. The bounds of this search space were chosen manually to ensure that higher held out log likelihood could not be achieved with a larger model.

## 4 Implementation

We implemented this model using `JAX` to parallelize the updates across mice and voxels. We fit the model on an NVidia A100 GPU. The algorithm takes between 5 minutes for the smallest models and 2 hours for the largest. These ranges include the time it takes to compile the algorithm.

Our code is open source and available at `https://github.com/lindermanlab/dirichlet-tucker`.

## References

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.