Supplementary Material: Effective Split-Merge Monte Carlo Methods for Nonparametric Models of Sequential Data

Michael C. Hughes[1], **Emily B. Fox**[2], and **Erik B. Sudderth**[1]

[1]Department of Computer Science, Brown University, {mhughes,sudderth}@cs.brown.edu
[2]Department of Statistics, University of Washington, ebfox@stat.washington.edu

Abstract

Here, we provide further details for the NIPS 2012 submission "Effective Split-Merge Monte Carlo Methods for Nonparametric Models of Sequential Data". The first section provides mathematics for sampling transition weights η, which is a correction from Fox et al. [1]. Next, we offer details on the procedure for split-merge updates. Finally, we provide additional experimental details and results for toy data experiments, analysis of motion capture sequences, and activity discovery using the CMU Kitchen videos.

A BP-HMM Transition Weights η Sampling Details

This section develops a corrected posterior distribution for sequence-specific transition weights η. Let η_i denote all transition weights for item i, and let η_{ij} be the (unnormalized) outgoing transition probability vector from state/behavior j in sequence i.

Fox et al. provide the following posterior for η_{ij}, which we find to be incorrect

$$\eta_{ijk}|\mathbf{z}_i \sim \text{Gamma}(N_{ijk} + \alpha + \delta_{j,k}\kappa, 1) \text{ (NOT CORRECT)} \tag{1}$$

where N_{ijk} counts the number of transitions from state j to k in sequence $\mathbf{z}_i$.

As a quick demonstration of incorrectness, note that the variance of the above distribution *increases* with more data (larger N_{ijk}), which is certainly not desirable.

Instead, the correct posterior for η_{ij} is given up to a proportionality constant as

$$p(\eta_{ijk}|\mathbf{z}_i, f_{ik} = 1) \propto \frac{(\eta_{ijk})^{N_{ijk}+\alpha+\delta_{j,k}\kappa-1} e^{-\eta_{ijk}}}{\left[\sum_{k':f_{ik'}=1} \eta_{ijk'}\right]^{N_{ij}}} \tag{2}$$

where N_{ijk} counts the number of transitions from state j to k in sequence $\mathbf{z}_i$, and $N_{ij} = \sum_{k:f_{i,k}=1} N_{ijk}$. This expression can be derived simply by multiplying the Gamma prior on η_{ijk} by the likelihood $p(\mathbf{z}_i|\eta_i, \mathbf{f}_i)$, and dropping all multiplicative factors constant with respect to η_{ijk}.

Draws from this posterior can be obtained by sampling auxiliary variables

$$\bar{\eta}_{ij} \sim \text{Dir}(\ldots, N_{ijk} + \alpha + \delta_{j,k}\kappa, \ldots) \tag{3}$$

$$C_{ij} \sim \text{Gamma}(K\alpha + \kappa, 1) \tag{4}$$

and then deterministically setting $\eta_{ijk} = C_{ij}\bar{\eta}_{ijk}$. This procedure inverts the usual Gamma to Dirichlet scaling transformation used to sample Dirichlet random variables.

As a sanity check, this correct procedure has variance that *decreases* with addition of more data. The Dirichlet draw $\bar{\eta}_{ij}$ will have smaller variance for entry k whenever N_{ijk} increases, and the C_{ij} scale factor is independent of observed data.

B BP-HMM Split-Merge MCMC Details

Here, we provide additional algorithmic details on the process of our novel split-merge moves. To review, our SM moves operate on the Markov state $\Psi = (\mathbf{F}, \mathbf{z})$ of the BP-HMM MCMC chain, where $\mathbf{F}$ denotes the binary feature assignment matrix and $\mathbf{z} = \{\mathbf{z}_1 \dots \mathbf{z}_N\}$ denote all HMM discrete state sequences. We will use $\mathbf{f}_i$ to denote the i-th row of $\mathbf{F}$.

First, we provide a high-level overview of a split-merge move in Alg. B.1. Next, we give details on the construction of a *merge* proposal in Alg. B.2, complementing the algorithm provided in the main paper (Alg. 1) for a *split* proposal. Finally, we provide a restricted Gibbs (RG) sampling algorithm (Alg. B.3) for drawing individual feature assignments k_a, k_b for one sequence within the sequentially allocated split proposal. This includes important details about how the transition probability is calculated for the acceptance ratio. Our hope is that this information might provide a knowledgable practioner sufficient details to fully understand and reproduce our split-merge MCMC moves.

B.1 Overall Split-Merge Procedure

As outlined in the main paper, completing one SM update requires several steps: selecting anchor items, selecting features to determine the appropriate move (split or merge), constructing the proposed assignment variables $\mathbf{F}^*, \mathbf{z}^*$ under the chosen move, and finally accepting or rejecting via Metropolis-Hastings. This high-level procedure is specified in Alg. B.1.

Alg. B.1 SplitMergeBPHMM($\mathbf{x}, \Psi, \alpha, \kappa, \lambda$)

Input: current markov state $\Psi = (\mathbf{F}, \mathbf{z})$
Output: new state $\Psi = (\mathbf{F}^*, \mathbf{z}^*)$
Procedure:

Select anchor items i, j uniformly at random from all data items
Select feature k_i at random from those possessed by item i
Select feature k_j according to $q(k_j|\mathbf{x}, \Psi, i, j, k_i)$ (see Eq. (5) of main paper)
if $k_i = k_j$ **then**
  $\Psi^*, \log q_{fwd} \leftarrow$ `SplitProposal(` $\Psi, \mathbf{x}, i, j, k_i, k_a, k_b$ `)`
  $\log q_{rev} \leftarrow$ `MergeProposal(` $\Psi^*, \mathbf{x}, i, j, k_a, k_b, k_i, \Psi$ `)`
else
  $\Psi^*, \log q_{fwd} \leftarrow$ `MergeProposal(` $\Psi, \mathbf{x}, i, j, k_i, k_j, k_m$ `)`
  $\log q_{rev} \leftarrow$ `SplitProposal(` $\Psi^*, \mathbf{x}, i, j, k_m, k_i, k_j, \Psi$ `)`
end if
Calc. joint log prob of current and proposed states $p(x, \Psi^*)$, $p(x, \Psi)$
Compute acceptance ratio ρ, depending on whether we split or merge

$$\rho = \frac{p(\mathbf{x}, \boldsymbol{\Psi}^*)}{p(\mathbf{x}, \boldsymbol{\Psi})} \frac{q_{rev}(\Psi|\Psi^*)}{q_{fwd}(\Psi^*|\Psi)} \frac{q_k(k_i, k_j|i, j, \Psi^*)}{q_k(k_i, k_j|i, j, \Psi)} \tag{5}$$

return Ψ^* with prob. $\min(\rho, 1)$, and Ψ otherwise

Eq. (5) provides the generic formula for computing the acceptance ratio of a split-merge move, agnostic to whether a split or a merge is actually proposed. To fill in details explicitly, the accept ratios for split and merge moves are as follows

$$\rho_{\text{split}} = \frac{p(\mathbf{x}, \mathbf{z}^*, \mathbf{F}^*)}{p(\mathbf{x}, \mathbf{z}, \mathbf{F})} \frac{q_{\text{merge}}(\mathbf{F}, \mathbf{z} \mid \mathbf{F}^*, \mathbf{z}^*, k_a, k_b)}{q_{\text{split}}(\mathbf{F}^*, \mathbf{z}^* \mid \mathbf{F}, \mathbf{z}, k_m)} \frac{q_k(k_a, k_b \mid \mathbf{F}^*, \mathbf{z}^*, i, j)}{q_k(k_m, k_m \mid \mathbf{F}, \mathbf{z}, i, j)} \tag{6}$$

$$\rho_{\text{merge}} = \frac{p(\mathbf{x}, \mathbf{z}^*, \mathbf{F}^*)}{p(\mathbf{x}, \mathbf{z}, \mathbf{F})} \frac{q_{\text{split}}(\mathbf{F}, \mathbf{z} \mid \mathbf{F}^*, \mathbf{z}^*, k_m)}{q_{\text{merge}}(\mathbf{F}^*, \mathbf{z}^* \mid \mathbf{F}, \mathbf{z}, k_a, k_b)} \frac{q_k(k_m, k_m \mid \mathbf{F}^*, \mathbf{z}^*, i, j)}{q_k(k_a, k_b \mid \mathbf{F}, \mathbf{z}, i, j)} \tag{7}$$

Next, we focus on the proposal construct procedures, denoted `MergeProposal` and `SplitProposal` in Alg. B.1. Note that the algorithm in the main paper specifies `SplitProposal`. Here, we provide the complementary sequential allocation procedure used to create a merge proposal.

B.2 Merge Proposal via Sequential Allocation

Constructing a merge proposal given input features k_a, k_b follows a sequential allocation process, similar to the `SplitProposal` outlined in the main paper (Alg. 1). Here, given fixed choice of k_a, k_b the proposed feature assignments are deterministic: we set $f^*_{\ell,k_m} = 1$ if ℓ is in the active set $\mathcal{S}$, and $f^*_{\ell,k_m} = 0$ otherwise. However, we still completely update the state sequences of the active set. Alg. B.2 specifies the necessary steps.

As in Alg. 1 of the main paper, the *order* in which the active set is traversed can be chosen uniformly at random. This choice has no bearing on the acceptance ratio, since it can be viewed as choosing a particular Markov transition kernel, just like the choice of anchor items i, j or the window location of the data-driven birth/death moves. All of these choices can be made *independent* of the current state of the sampler $(\mathbf{F}, \mathbf{z})$, and thus need not be considered in the acceptance ratio.

Alg. B.2 MergeProposal($\Psi, \mathbf{x}, i, j, \mathcal{S}, k_a, k_b, k_m$)

Input: current MCMC chain state $\Psi = (\mathbf{F}, \mathbf{z})$
Output: new state $\Psi^* = (\mathbf{F}^*, \mathbf{z}^*)$
Procedure:

1: Delete assignments to old features: $\mathbf{F}^* \leftarrow \mathbf{F}$, $\mathbf{F}^*_{:,[k_a k_b]} \leftarrow [0 \quad 0]$
2: Add new feature k_m to every item $\ell \in \mathcal{S}$: $\mathbf{f}^*_{\ell,k_m} \leftarrow 1$
3: Initialize proposed state sequences, setting anchor state sequences to new feature k_m

$$\begin{aligned} z^*_{\ell t} &\leftarrow z_{\ell t} \ \text{ for all items } \ell \\ z^*_{it} &\leftarrow k_m \ \text{ if } z_{it} = k_a \text{ or } z_{it} = k_b \\ z^*_{jt} &\leftarrow k_m \ \text{ if } z_{jt} = k_a \text{ or } z_{jt} = k_b \end{aligned} \tag{8}$$

4: Create emission parameters $\hat{\theta} \leftarrow$ posterior mean of $p(\theta|\mathbf{x}, \mathbf{z}^*)$ for all features
5: Create transition parameters $\hat{\eta}_\ell \leftarrow$ prior mean of η_ℓ for all items $\ell \in \mathcal{S}$
6: **for all** non-anchor items ℓ in active set $\mathcal{S}$ **do**
7: sample state sequence $\mathbf{z}^*_\ell \sim p(\mathbf{z}_\ell|\mathbf{x}_\ell, \mathbf{f}_\ell, \hat{\theta}, \hat{\eta}_\ell)$
8: update emission parameters $\hat{\theta}_{k_m} \leftarrow$ posterior mean of $p(\theta_{k_m}|\{x_{nt}|z^*_{nt} = k_m, n \in \mathcal{S}_{prev}\})$
9: *NB: This expectation is only taken w.r.t. sequences $\mathbf{x}_\ell, \mathbf{z}^*_\ell$ already visited in the active set*
10: **end for**
11: Sample for anchor items i, j
12: sample state sequence $\mathbf{z}^*_i \sim p(\mathbf{z}_i|\mathbf{x}_i, \mathbf{f}_i, \hat{\theta}, \hat{\eta}_i)$
13: sample state sequence $\mathbf{z}^*_j \sim p(\mathbf{z}_j|\mathbf{x}_j, \mathbf{f}_j, \hat{\theta}, \hat{\eta}_j)$

B.3 Details of Restricted Gibbs sampling

In Alg. B.3 below, we show in detail the procedure for resampling the feature assignments for one particular data item, as in line 5 of Alg. 1 in the main paper. Note that we only sample the item's assignment to the two features created by a split move, and we only consider candidate assignments in which *at least one* of new features is possessed by the object. This restriction maintains reversibility of the split-merge move.

$$f_{\ell,[k_a\ k_b]} \sim \begin{cases} [1\ 0] \\ [0\ 1] \\ [1\ 1] \end{cases} \text{ w. prob. } \quad \propto p(f_{\ell,[k_a\ k_b]}|f_{\mathcal{S}_{prev},[k_a\ k_b]})p(\mathbf{x}_\ell|f_\ell, \eta_\ell, \hat{\theta}) \tag{12}$$

This is a restricted Gibbs (RG) algorithm that procedes by sequential allocation: we sweep through the active set $\mathcal{S}$ and allocate a restricted set of features k_a, k_b to item ℓ based on the assignments

Alg. B.3 SampleSplitFeatures-RestrictedGibbs($\ell, \mathbf{F}, \hat{\theta}, \hat{\eta}, \mathbf{x}_\ell, \mathcal{S}_{\text{prev}}, [k_a, k_b], \mathbf{F}^{\text{target}}$)

Input: feature matrix $\mathbf{F}$, HMM parameters $\hat{\theta}, \hat{\eta}$, data $\mathbf{x}_\ell$, set of previously updated items $\mathcal{S}_{\text{prev}}$
OPTIONAL: desired output $\mathbf{F}^{\text{target}}$ (for computing reverse probability of a merge move)
Output: (1) updated feature assignments $f_{\ell,[k_a,k_b]}$ for item ℓ , given $\mathbf{F}_{\mathcal{S}_{\text{prev}},[k_a\ k_b]}$
note that this new assignment is restricted: item ℓ must possess at least one of k_a, k_b
(2) prob. of transition from input f_ℓ to new assignment, denoted by q_{RG}
Procedure:

1: Define m_a, m_b as counts of how many previously seen items (not ℓ) possess features k_a, k_b.

$$m_a = \sum_{a \in \mathcal{S}_{\text{prev}}} f_{a,k_a} \qquad m_b = \sum_{b \in \mathcal{S}_{\text{prev}}} f_{b,k_b} \qquad N_{\text{prev}} = |\mathcal{S}_{\text{prev}}| \tag{9}$$

2: Construct Gibbs conditional probabilities for each possible value of $f_{\ell,[k_a k_b]}$
Note: only defined here up to a multiplicative constant

$$\begin{aligned} p_{[1\ 0]} &\leftarrow (\quad m_a \quad)(N_{\text{prev}} - m_b + \beta) p(x_\ell | f_{\ell,[k_a\ k_b]} = [1\ 0], \hat{\eta}_\ell, \theta) \\ p_{[0\ 1]} &\leftarrow (N_{\text{prev}} - m_a + \beta)(\quad m_b \quad) p(x_\ell | f_{\ell,[k_a\ k_b]} = [0\ 1], \hat{\eta}_\ell, \theta) \\ p_{[1\ 1]} &\leftarrow (\quad m_a \quad)(\quad m_b \quad) p(x_\ell | f_{\ell,[k_a\ k_b]} = [1\ 1], \hat{\eta}_\ell, \theta) \end{aligned} \tag{10}$$

3: Compute normalization constant for the choice of feature assignment:

$$Z_p \leftarrow p_{[1\ 0]} + p_{[0\ 1]} + p_{[1\ 1]} \tag{11}$$

4: **if** $\mathbf{F}^{\text{target}}$ exists: **then**
5: $\quad f_{\ell,[k_a\ k_b]} \leftarrow f^{\text{target}}_{\ell,[k_a\ k_b]}$
6: **else**
7: $\quad f_{\ell,[k_a\ k_b]} \sim \begin{cases} [1\ 0] & \text{with prob. } \frac{p_{[1\ 0]}}{Z_p} \\ [0\ 1] & \text{with prob. } \frac{p_{[0\ 1]}}{Z_p} \\ [1\ 1] & \text{with prob. } \frac{p_{[1\ 1]}}{Z_p} \end{cases}$ *Draw k_a, k_b feature assignment for item ℓ*
8: **end if**
9: $q_{RG} \leftarrow \begin{cases} \frac{p_{[1\ 0]}}{Z_p} & \text{if } f_{\ell,[k_a\ k_b]} = [1\ 0] \\ \frac{p_{[0\ 1]}}{Z_p} & \text{if } f_{\ell,[k_a\ k_b]} = [0\ 1] \\ \frac{p_{[1\ 1]}}{Z_p} & \text{if } f_{\ell,[k_a\ k_b]} = [1\ 1] \end{cases}$ *Record prob. transition from old to new f*

made to previously visited sequences. We call this set of previously assigned items $\mathcal{S}_{\text{prev}}$, which is a subset of $\mathcal{S}$. Its size is $N_{prev} = |\mathcal{S}_{\text{prev}}|$

Studying the posterior equation for $f_{\ell,[k_a\ k_b]}$, we can see that it nicely factors into a prior and a likelihood. The prior term can be computed using the Indian Buffet process generative model, since both k_a and k_b are shared features. Thus, it is simply ratios of count sufficient statistics. The probability that we possess k_a is just m_a (the number of previous sequences that possess k_a) divided by $N_{prev} + \beta$. The likelihood term is simply the marginal probability of the observed sequence, summing over all possible discrete state sequences, given the deterministic auxiliary parameters $\hat{\theta}, \hat{\eta}_\ell$. This considers using both the proposed available features k_a, k_b as well as all other features possessed by item ℓ. This likelihood can be computed efficiently via dynamic programming, and this exact routine is conveniently also necessary for the standard sampling of shared features.

Two things are important to highlight about this algorithm as a piece of a larger split-merge proposal move. First, we must track the proposal probabilities q_{RG} of making each random choice. These are used in the acceptance ratio of Alg. B.1. Second, we must be able to compute the probability of *reverse* moves. That is, if we perform a merge from Ψ to Ψ^*, the acceptance ratio requires computing the probability of obtaining Ψ from Ψ^* via a split. In this setting, we know in advance the destination state Ψ, so we don't actually need to sample anything, but we do need to walk-through the sampling process step by step and tally the probability of making each discrete choice that leads from Ψ^* back to Ψ. When the split construction process is called with a known destination state $\Psi^{\text{target}} = (F^{\text{target}}, \mathbf{z}^{\text{target}})$, the procedure does not randomly assign variables, but instead computes

the probability of assigning to the provided target values. Although Alg. B.3 describes the direct calculation of these transition probabilities q_{RG}, in practice we recommend only tracking logarithms of this quantity to prevent numerical problems.

Note that an analogous process to B.3 occurs for resampling the state sequence $\mathbf{z}_\ell$ for some item ℓ in both split and merge moves. A total probability q_z of transitioning from $\mathbf{z}$ to $\mathbf{z}^*$ is computed by tallying up the transition probabilities at each discrete assignment made by the Gibbs block sampler of $\mathbf{z}_\ell$ given fixed $\mathbf{f}_\ell$ and point estimates $\hat{\theta}, \hat{\eta}_\ell$. When computing reverse moves, a target configuration is provided, and rather than actually making random assignments the algorithm computes the probability of the sequence of choices that leads to the target state.

C Toy Data Experimental Details

Here, we provide further details for the toy data experiments from Sec. 4-1 of the main paper.

C.1 Hyperparameter Settings

For all toy data experiments, we fix BP hyperparameters to $\gamma = 2, \beta = 1$ and HMM hyperparameters to $\alpha = 2, \kappa = 200$ (which yields high stickiness to match the high self-transition probability used to generate the data). We could have easily sampled these hyper parameters, but we wish to directly compare the inference algorithms on $\mathbf{F}, \mathbf{z}$ without extra complications. Additionally, we note that when initialized to have just one state shared by all sequences, the parameters α, κ no longer are identifiable (since just one state means there are no transitions possible, and thus the "sticky" parameter is meaningless). To prevent complications, we just fixed these to reasonable values for these experiments.

C.2 Remove Redundant Feature Experiments

As mentioned briefly in the main paper, we study the ability of various BP-HMM samplers to *remove* redundant features via a `repeat` initialization, diagrammed in Fig. C.1. The data is generated by 8 true states, and we know the actual $\mathbf{F}, \mathbf{z}$ values for each generated item. We then create two "copies" of each of the 8 true states, so that there are 16 states total, containing two disjoint sets of behaviors sufficient to explain all the data. We assign the first half of data to set 1, and the second half to set 2, initializing both $\mathbf{F}$ and $\mathbf{z}$ to their "true" values within the appropriate set. This creates a challenging scenario for any inference method, since it must successfully merge down to one set of true states, when each sequence is already stuck in a local optimum.

We show trace plots of all sampler methods in Fig. C.2. We observe that for Gaussian and AR likelihoods, the SM moves quickly converge down to a mode with all redundant states removed, sometimes within five minutes. In contrast, both DD and Prior reversible jump proposals never merge down to the 8 true features within the alotted hour. For Multinomial likelihoods, SM moves are still faster, but the margin of victory, while clearly noticeable, is somewhat smaller. The baseline samplers with Gibbs and reversible jump moves are able to delete down to 8 features eventually in this scenario.

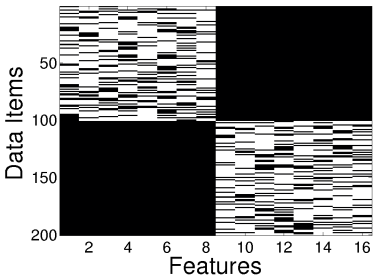

Figure C.1: Feature matrix for a `repeat` initialization. White indicates feature presence.

We attribute this faster convergence to the particular multinomial toy data under study: each time step emitted on average 5 symbols out of a vocabulary of 500, so parameter estimates for each copy of a true state are likely to be noisy and slightly different depending on which symbols happen to be seen more often. This noisyness can easily cause one copy to be preferred over another after a while, allowing the Gibbs moves to converge down to the true posterior with 8 states. We caution that it is difficult to compare across likelihoods in general, since different settings of the likelihood hyperparameters can cause very dif-

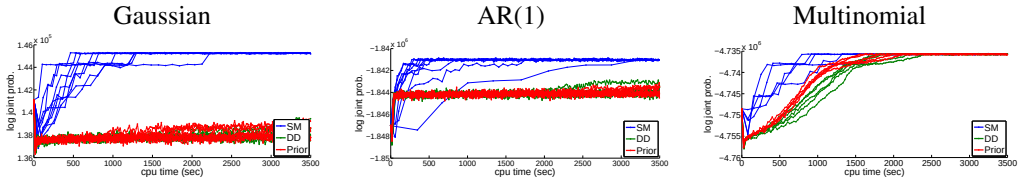

Figure C.2: Log probability trace plots comparing inference methods on the "remove redundant features" task, from Sec. 4.1 of the main paper.

.

ferent behavior of the standard Gibbs methods, and we did not carefully match these settings across datasets.

Overall, we observe that our SM moves make significant improvements over more local samplers in this challenging setting, making a strong case for our new inference methods.

D Motion Capture Experimental Details

Here, we provide further details on the analysis of the motion capture (mocap) sequences, from Sec. 4.2 of the main paper.

D.1 Data Preprocessing

We use the same 12 channels as in [1], corresponding (roughly) to measurements of torso, neck, R/L hips, R/L shoulders, R/L wrist, R/L knees, and R/L feet. All sequences in all datasets (small and large) are recorded at 120 fps, and we take block averages with no overlap to obtain a $D = 12$ dimensional vector at each 0.1 second interval of motion.

D.2 Hyperparameter Settings

As in [1], we *resample* hyperparameters HMM parameters α, κ and BP parameter γ at each iteration of all sampler runs on motion capture data. We further resample the concentration parameter β, using Metropolis-Hastings updates with a Gamma proposal centered on the previous value for β, similar to the proposals [1] discusses for α, κ.

For the prior on the parameters of the first-order auto-regressive Gaussian likelihood for the observed joint angles, we choose the matrix-normal inverse Wishart due to conjugacy. This prior requires several hyperparameters: scale matrix S_0, degrees of freedom ν_0, and column-wise precision matrix R_0. Together, these specify a prior on autoregressive coefficients A and covariance Σ for a first-order AR process with $D = 12$ dimensional observations as follows:

$$\Sigma \sim \mathcal{W}_D^{-1}(\nu_0, S_0) \tag{13}$$

$$A|\Sigma \sim \mathcal{MN}_D(\mathbf{0}, \Sigma, R_0) \tag{14}$$

We set $\nu_0 = D + 2$, S_0 to 0.5 times the empirical covariance of first differences of all observation sequences, and $R_0 = 0.5 I_D$, where I is the identity matrix in D dimensions. We note that [1] set S_0 to 5 times the empirical covariance, rather than 0.5 as we do. We made this choice because with our improved inference, we need not specify such a vague prior. Realistically, *individual* discovered behaviors should have less variability than all motion sequences taken together without regard for different actions. We validated our better setting of these parameters by performing a "cheating" initialization of an MCMC run: starting from the human annotated ground truth of the small 6 sequence dataset, we found that under our settings the sampler stayed near the "ground truth" configuration throughout thousands of iterations. In contrast, a sampler initialized to truth under the more vague prior of [1] quickly wandered away from this ground truth, gaining in likelihood by deleting several unique behaviors. When the goal is comparing sampler runs to this ground truth, we found it better to use hyperparameter settings in which ground truth is a (possibly local) mode of the posterior.

Note that this difference in likelihood parameter settings might explain why in our experiments the BPHMM sampler from the *prior* performs somewhat worse (in terms of Hamming distance) on the 6 sequence dataset than reported in [1].

E CMU Kitchen Experimental Details

Here, we provide further details on the analysis of the CMU Kitchen dataset, from Sec. 4.3 of the main paper.

E.1 Data Preprocessing: Quantization of HOG/HOF descriptors at sparse interest points

We follow the same preprocessing steps as [2] to obtain a "bag-of-features" time series representation of the Kitchen videos. We use existing *spatio-temporal interest points* (STIP) code [3] to detect interest points and obtain *histogram of gradients* (HOG) and *histogram of optical flow* (HOF) descriptors, which we concatenate together. For each version of the Kitchen dataset (*small*: 30 videos of 3 recipes, or *big*: 126 videos from 5 recipes), we build a codebook from these descriptors with $V = 1000$ codewords using the K-means algorithm. Each STIP is then mapped to the nearest codeword, providing a standard "bag of words" representation [4]. To represent videos as discrete time series, we choose a temporal bin-width w (in seconds for invariance to frame rate), divide video i into T_i bins of width w, and count the number of occurrences of each codeword across all STIPs within each bin. The parameter w indirectly influences the time-scale of the learned dynamics.

We set $w = 0.5$ seconds on the small dataset, to match the previous settings in [2]. On the big dataset, we set $w = 1$ second, which we find to be more appropriate timescale for the behaviors of interest, since each video is several minutes long.

E.2 Hyperparameter Settings

As in [2], we find that fixing hyperparameters provides reasonable performance for our inference algorithm. We set BP hyperparameters $\gamma = 2$ and $\beta = 1$, to encourage moderate sharing and limit the creation of too many states. Without setting γ small, we find the sampler creates an over-abundance of states that does not lead to sensible human interpretation. We further set HMM parameters to $\alpha = 2$ and $\kappa = 10\alpha$, which leads to reasonable stickyness in behavior transitions.

For the small dataset of 30 videos shown in Fig. 5, we set the hyperparameter of the Dirichlet prior on multinomial emissions θ to $\lambda = 1$ (similar to [2]'s 0.75). On the large dataset, we find that the quantized multinomial likelihood leads to an over-abundance of behaviors with $\lambda = 1$, and so we set $\lambda = 10$ to encourage the model to find a *compact* set of behaviors, which still ends up containing 73 features in all.

E.3 Key Frames of Activity Discovery on 126 Kitchen Sequences

Fig. E.1 shows example frames from 8 representative behaviors found by our SM+DD inference algorithm, out of the total collection of 73 behaviors. Each key frame is taken from a different sequence. We selected key frames from the longest contiguous segments assigned to each behavior, filtering out segments with no observed STIPs in more than half of all timesteps.

These key frames indicate that the sampler does find coherent and intuitive behaviors, despite the fact the STIPs have well-known weaknesses as video representation [2]. Interestingly, the sampler finds at least two distinct styles of stirring, which are used most often in Brownie videos but also sometimes in Eggs videos. Examining the data indicates that these motions are truly different: as in [2], these behaviors seem to correspond to different styles of stirring the bowl. Thus, we should not expect our SM moves to merge these into one behavior.

E.4 Commentary on Behavior Assignment Map (Fig. 6 in main paper)

Fig. 5 of the main paper shows the usage patterns of a handful of manually selected representative behaviors over time across all videos in the corpus. To make this figure, we discretize each video into 50 blocks, and count the number of timesteps within each block assigned to each feature. If a

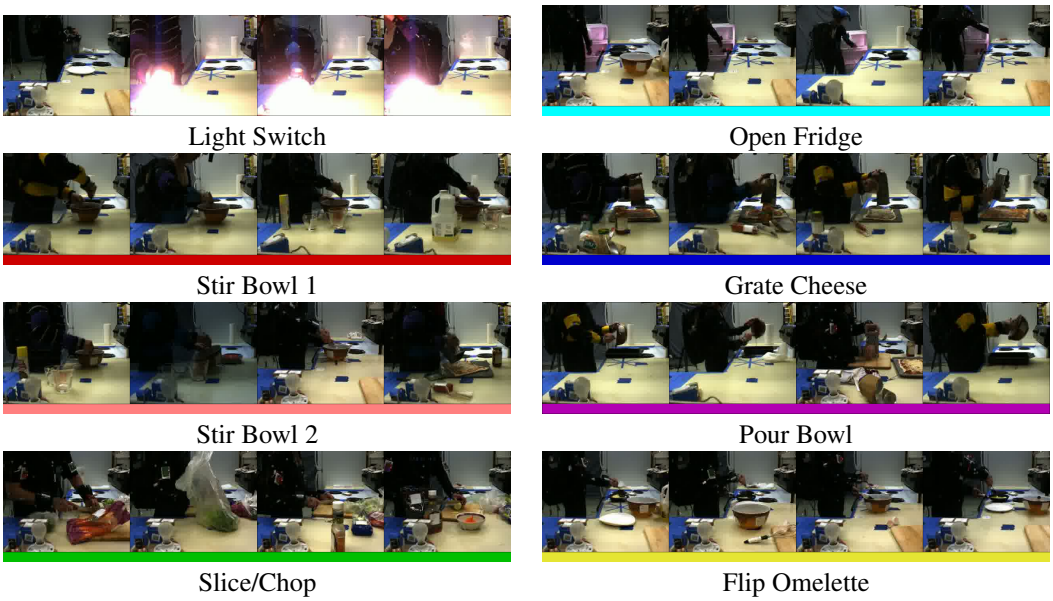

Figure E.1: Qualitative analysis of 126 Kitchen vides: 8 representative behaviors from the set of 73 recovered by the BP-HMM with SM+DD moves. Best viewed in color on a computer screen.

behavior has at least two detections in a particular block, we fill in that block with the appropriate color. We disregard timesteps that do not have any observed STIPs, as these assignments are not likelihood-driven and thus not informative.

A careful reader will observe that within each recipe category of Fig. 5, some behavior usage is not uniformly distributed. Both "flip omelette" and "chop vegetables" behaviors occur mostly in the bottom half of the displayed listing of Eggs and Salad videos respectively. Examining the data indicates that this usage pattern is particular to the way in which this data was originally collected. The "flip omelette" behavior is exclusive to sequences in which actors use a spatula (rather than a fork) to perform the flipping. This tool likely has more distinct motions and more distinct HOG gradient features. The "slice/chop" behavior is particular to sequences in which actors use a large knife and cut carrots and cucumbers on a cutting board, while some earlier trials had actors just slicing in their hands with a butter knife. Inspection indicates that spatula and cutting board tools were more readily available in motion capture trials collected at later dates, perhaps because instructions to subjects were revised to encourage use of better tools. This explains the non-uniformity in usage within Fig. 5, since we display each category in increasing order of subject ID number, which is directly related to time of collection.

References

[1] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Sharing features among dynamical systems with beta processes. In *NIPS*, 2010.

[2] M. C. Hughes and E. B. Sudderth. Nonparametric discovery of activity patterns from video collections. In *CVPR Workshop on Perceptual Organization in Computer Vision*, 2012.

[3] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[4] H. Wang, M.M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.