Proceedings Paper:
Wang, H orcid.org/0000-0002-2281-5679, Ondřej, J and O'Sullivan, C (2016) Path Patterns: Analyzing and Comparing Real and Simulated Crowds. In: Wyman, C, Yuksel, C and Spencer, SN, (eds.) Proceedings. I3D '16: 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 26-28 Feb 2016, Redmond, WA, USA. ACM , pp. 49-57. ISBN 978-1-4503-4043-4

https://doi.org/10.1145/2856400.2856410

Supplementary Material

Terms	Notation	Meaning
Agent State	w	$w = \{p, v\}$ where p and v are the position and orientation of an agent
Data Segments	d	Trajectory and velocity data of the crowd. $d = \{w_i\}$.
Path Pattern	β	A mixture of paths.
DP Atoms	h_{l_i}, c_{d_j}	
DP Weight Parameters	$v_k, g_l, \epsilon_{l_i}, \pi_{d_j}$	
Dirichlet Parameter	η	
Beta Parameter	a, b, ω, α	
Component Indices	j, i, l, k and their totals: J, I, L, K	

Table 1: *Terminology and Parameters*

1 SV-DHDP Model

We first briefly review Dirichlet Processes (DPs) and Dependant Dirichlet Processes (DDPs). A DP can be seen as a probabilistic distribution over distributions, which means any draw from a DP is a probabilistic distribution itself. In a *stick-breaking* representation [Sethuraman 1994] of DP: $G = \sum_{k=1}^{\infty} \sigma_k(v)\beta_k$, where $\sigma_k(v) = v_k \prod_{j=1}^{k-1}(1 - v_j)$, $\sum_{k=1}^{\infty} \sigma_k(v) = 1$ and $\beta_k \sim H(\eta)$.

β_k are DP *atoms* drawn from some base distribution H. In our problem, they are Multinomials drawn from a $Dirichlet(\eta)$. The $\sigma_k(v)$s are called *stick proportions* or DP weights because it mimics breaking a stick iteratively in the following way. Assuming the length of a stick is 1, in each iteration, a proportion v_k of what is left of the stick , $\prod_{j=1}^{k-1}(1 - v_j)$, is broken away. v is Beta-distributed.

A DDP [MacEachern 1999] generalizes the concept of DP by replacing its weights and atoms with stochastic processes. In our context, a DDP can be represented as: $G_l = \sum_{i=1}^{\infty} \sigma_i(v) G_{l_i}$, where everything is the same as the DP representation except that the atoms are now $\{G_{l_i}\}$. Each G_{l_i} itself is a DP and all $\{G_{l_i}\}$ are draws from same base DP. Both DP and DDP are ideal priors for modeling infinite clusters.

With terminologies defined in Table 1 and equiped with DP and DDP, we are ready to fully define our model. In a standard hierarchical Bayesian setting, a tree is constructed in attempt to explaining the observations through a hierarchy of factors. In our problem, the observations are *agent states*. We segment them into equal-length *data segments* along time domain. Our goal is to find a set of *path patterns* $\{\beta_k\}$ that, when combined with their respective weights, best describe all the segments in terms of their likelihoods.

As shown in the toy example in the paper, a subset of $\{\beta_k\}$ is needed to describe a *data segment*. $\{\beta_k\}$ are shared across all segments. A two-layer tree is used to model this phenomenon. The root node is $\{\beta_k\}$ governed by a global DP prior. Each leaf node represents a *data segment* with a DP drawn from the global DP prior to model its own pattern set $\{\beta_k\}_d \subset \{\beta_k\}$. This is a standard two-layer HDP.

Further, imagine some *data segments* share a bigger subset of $\{\beta_k\}$, namely $\{\beta_k\}_c$, so that $\{\beta_k\}_d \subset \{\beta_k\}_c \subset \{\beta_k\}$ and they form a *segment cluster*. Also, we have potentially infinitely many such clusters. We need a middle layer to capture this effect. At this layer, there is a nested clustering. First, each $\{\beta_k\}_c$ can contain infinite elements. Second, the number of clusters can be infinitely big. This effect can be captured by adding a DDP layer immediately below all $\{\beta_k\}$ but higher than leaf nodes. After constructing such a tree structure, we can compute $\{\beta_k\}$ by clustering the agent states layer by layer up to the top.

Such a tree structure is shown in Figure 1. Each sharp-cornered rectangle is a DP. G on the right is the global DP over $\{\beta_k\}$. The bottom-left segment-level distribution, G_d, is the local DP over $\{\beta_k\}_d$. G_l is the DDP. The number of atoms in G_l is the number of segment clusters. Each atom G_{l_i} is a DP governing $\{\beta_k\}_c$ All stick proportions sum to 1, $s.t.$ $\sum v = 1$, $\sum g = 1$, $\sum \epsilon = 1$ and $\sum \pi = 1$. β_k, h_{l_i} and c_{d_j} are DP atoms.

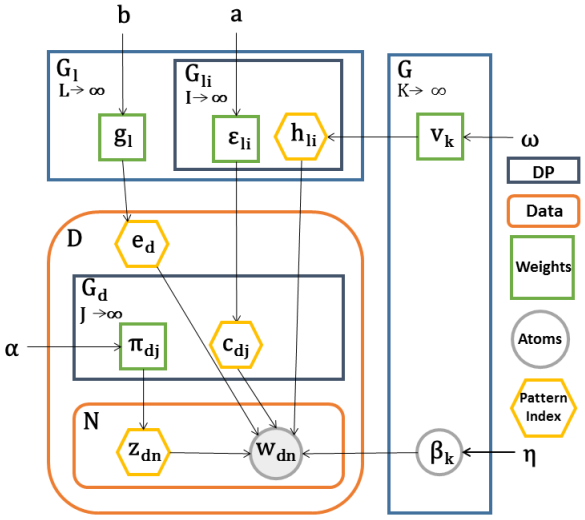

Figure 1: *SV-DHDP model. Sharp-corner rectangles (G, G_d, G_{l_i} and G_l) are DPs in which squares are weights and circles are atoms. Rounded rectangles are data samples or data segments. Hexgons are pattern assignments. N is the number of agent states in segment D.*

This model explains how the observations, w (shaded), are generated from β_k through a hierarchical factors between w_n and β_k. This dependency is explained in Algorithm 1 in the supplementary material. We explain, in Algorithm 1, the dependency between the observed agent states and the latent path patterns we are solving for.

2 Variational Inference for SVDHDP

In this section we give details of the variational inference of our SVDHDP model. Variational Inference (VI) [Bishop 2007] approximates a target distribution by solving an optimization problem. When the target distribution is intractable, VI uses a family of tractable distributions (variational distributions) to approximate the target distribution. By optimizing for the parameters of the variational distributions, the target distribution can be approximated. The optimization is done by minimizing the Kullback-Leibler (KL) divergence between the posterior distribution and the variational distribution $q(\beta, \Omega)$, which amounts to maximizing the

- For the global DP, build G by:
- Drawing an infinite number of patterns, $\beta_k \sim$ Dirichlet(η) for k$\in\{1, 2, 3, \dots\}$
- Drawing the stick proportions, $\sigma_k(v)$ where $v_k \sim$ Beta(1, ω) for k$\in\{1, 2, 3, \dots\}$
- For each segment cluster l, build a G_l by:
  1. Drawing stick proportions, $g_l \sim$ Beta(1, b) for $l \in\{1, 2, 3, \dots\}$
  2. Since the atoms of G_l are DPs, build each atom G_{l_i} by:
     (a) Drawing an infinite number of pattern indices, $h_{l_i} \sim$ Multinomial($\sigma(v)$) for i$\in\{1, 2, 3, \dots\}$
     (b) Drawing the stick proportions, $\sigma_i(\epsilon)$, where $\epsilon_{l_i} \sim$ Beta(1, a) for i$\in\{1, 2, 3, \dots\}$
     (c) For each data segment d, build G_d by:
        i. Drawing an infinite number of data segment cluster indices, $e_d^l \sim$ Multinomial($\sigma(g)$) for l$\in\{1, 2, 3, \dots\}$
        ii. Drawing an infinite number of group pattern indices, $c_{d_j} \sim$ Multinomial($\sigma(\epsilon_l)$) for j$\in\{1, 2, 3, \dots\}$
        iii. Drawing the stick proportions, $\sigma_j(\pi)$, where $\pi_{d_j} \sim$ Beta(1, α) for j$\in\{1, 2, 3, \dots\}$
        iv. For each data sample w:
           A. Draw a pattern assignment, $z_{d_n} \sim$ Multinomial($\sigma(\pi)$).
           B. Generate a data sample $w_n \sim$ Multinomial(β_{h_u}), where u = e_x, $x = c_y$ and y = z_{d_n}.

Algorithm 1: Data sample generation in SV-DHDP. Dirichlet, Beta and Multinomial stand for their eponymous distributions.

evidence lower bound(ELBO), a lower bound on the logarithm of the marginal probability of the observations $log\ p(w)$:

$$\mathcal{L}(q) = \mathbb{E}_q[log\ p(w, \beta, \Omega)] - \mathbb{E}_q[log\ q(\beta, \Omega)] \tag{1}$$

The mean-field family is the simplest for approximating the posterior. It assumes each model parameter is only conditioned on its own hyper-parameters:

$$q(\beta, \Omega) = \prod_{k=1}^{K} q(\beta_k|\lambda_k)(\prod_{m=1}^{M}\prod_{r=1}^{R} q(\Omega_{mr}|\xi_{mr})) \tag{2}$$

where λ is the parameter governing the distribution of the global parameter. Parameter ξ_{mr} governs the distribution of the local parameter Ω_{mr} in the mth context (e.g., the mth data segment or the mth cluster). Here, M and R do not have specific meanings and are only for illustration purpose.

We then optimize Equation 1 for λ and ξ. Since all the distributions in SV-DHDP are from the exponential family, we assume that $q(\Omega|\xi)$ and $q(\beta|\lambda)$ are also from the exponential family which has the general form:

$$p(\beta|w, \Omega, \eta) = h(\beta)exp\{\rho_g(w, \Omega, \eta)^T t(\beta) - a_g(\rho_g(w, \Omega, \eta))\} \tag{3}$$

where scalar functions $h()$ and $a()$ are $base\ measure$ and $log - normalizer$; the vector functions $\rho()$ and $t()$ are the $natural\ parameter$ and $sufficient\ statistics$. For optimizing Equation 1 with respect to λ, we take the gradient:

$$\nabla_\lambda \mathcal{L} = \nabla_\lambda^2 a_g(\lambda)(\mathbb{E}_q[\rho_g(w, \Omega, \eta)] - \lambda) \tag{4}$$

and we can set it to zero by setting:

$$\lambda = \mathbb{E}_q[\rho_g(w, \Omega, \eta) - \lambda] \tag{5}$$

The optimization for ε is similar to Equation 5.

2.1 Natural Gradient

Since we are trying to optimize the parameters to minimize the KL-divergence, it is more reasonable to compute the $natural\ gradient$ of the ELBO instead of the Euclidean gradient. The natural gradient of a function accounts for the information geometry of its parameter space, using a Riemannian metric to correct the traditional gradient. According to [Amari 1998], a natural gradient can be computed by pre-multiplying the gradient by the inverse of the Riemannian metric $G(\omega)^{-1}$:

$$\hat{\nabla}_\lambda f(\lambda) \triangleq G(\lambda)^{-1} \nabla f_\lambda(\lambda) \tag{6}$$

where $G(\lambda)^{-1}$ is the Fisher information matrix of $q(\lambda)$. When $q(\beta|\lambda)$ is from the exponential family, $G(\lambda) = \nabla_\lambda^2 a_g(\lambda)$ and $\hat{\nabla}_\lambda \mathcal{L} = \mathbb{E}_\epsilon[\rho_g(w, \Omega, \eta)] - \lambda$. The natural gradient of $\mathcal{L}$ with respect to ξ is in a similar form, but only depending on its local contexts.

2.2 Stochastic Optimization

Optimizing Equation 1 for λ and ξ by a traditional coordinate ascent algorithm involves nested iteration loops. The inner loop iterates on all data segments to update ξ until it converges and jumps out to the outer loop to make one update on λ, then the iteration starts over again until λ also converges. This is very slow especially when the number of data segments is large, because before updating λ for one step, the inner loop has to compute the gradient at every data segment in the dataset.

To further speed up the training, we employ Stochastic Optimization. Stochastic optimization uses noisy gradient estimates with a decreasing step size to discover good local optima. Noisy gradients are usually cheaper to compute and help avoid low quality local optima. With certain conditions on the step size, it provably converges to an optimum [Robbins and Monro 1951]. Stochastic optimization uses a noisy gradient distribution $B(\lambda)$ so that $\mathbb{E}_q[B(\lambda)] = \nabla_\lambda f(\lambda)$. It allows us to update λ:

$$\lambda^{(t)} = (1 - \rho_t)\lambda^{(t-1)} + \rho_t b_t(\lambda^{(t-1)}) \tag{7}$$

where b_t is an independent draw from the noisy gradient B, t is time step and the step size ρ_t satisfies:

$$\sum \rho_t = \infty; \sum \rho_t^2 < \infty \tag{8}$$

Specifically, we use:

$$\rho_t = (t + \tau)^{-\kappa} \tag{9}$$

where τ down-weights the early iterations and κ, the forgetting rate, controls how much the new information is valued in each iteration. From Equation 7, we can sample the gradient on one data segment instead of all of them to compute the gradient.

We further extend Equation 7 to a $mini\ batch$ version of Equation 7. In each iteration, we sample D data segments and compute

Equation 7 for each of them, then average the results as the final update:

$$\lambda^{(t)} = (1-\rho_t)\lambda^{(t-1)} + \rho_t \frac{1}{D}\sum_d b_d^t(\lambda^{(t-1)}) \tag{10}$$

where $b_d^t()$ is the stochastic gradient computed from sample d and D is the sample number. Since the mini batch version is highly parallelizable and gives better estimations of the gradient, we thus further speed up the computation and improve the results.

In practice, we cannot perform computations for an infinite number of path patterns. So a $truncation\ number$ is given at each level. This number is the maximum cluster number modeled at its level. It is set bigger than needed so that only a part of clusters are used in the clustering. The truncation number for each layer is much smaller than the one above it because we expect a much smaller number of path patterns in a child node than its parent. We emphasize that this is fundamentally different from giving a pre-defined cluster number and the model can still automatically compute the desirable number of clusters.

Given, $\mathbf{D}$ data segments, each containing $\mathbf{N}$ agent states, we assume that the whole data set contains k path patterns where $k < \mathbf{K}$. Data segments can be clustered into l clusters where $l < \mathbf{L}$, each of which contain i path pattern indices where $i < \mathbf{I}$. Finally, in each data segment $\mathbf{d}$, the agent states can be clustered into j groups where $j < \mathbf{J}$ groups. We give the overall algorithm in Algorithm 2 and refer the readers to the supplementary material for the function subroutines and the mathematical deduction.

Algorithm 2: VI Optimization

Initialize λ^0, set o^1 = 1 and $o^2 = \omega$, p^1 = 1, p^2 = b, q^1 = 1, q^2 = a;
Set up step size ρ_t, set init t = 0;
while *not converged* **do**
  sample a data segment w_d;
  [$\varepsilon\prime$, μ_d, ζ_d, ϕ_d] = $\mathbf{initLocal}$(w_d, λ) (Algorithm 3);
  [μ_d, ζ_d, ϕ_d] = $\mathbf{opLocal}$(w_d, ε, ζ_d, ϕ_d, β, g) (Algorithm 4);
  [ϵ, ζ_d, ϕ_d] = $\mathbf{updateCluster}$(w_d, ε, ζ_d, ϕ_d, v, β) Algorithm 5);
  [$\lambda^{(t)}$, $o^{1(t)}$,$o^{2(t)}$] = $\mathbf{updateGlobal}$(w_d, η, ε, ζ_d, ϕ_d, ρ^t, $\lambda^{(t-1)}$, $o^{1(t-1)}$, $o^{2(t-1)}$) (Algorithm 6);
  t = t + 1;
  update ρ with t (Equation 9);
end

2.3 Computational Details

Based on Figure 1 and the complete conditional in explained in the paper Equation 2. However, Equation 2 is for the purpose of explaining Variational Inference in the paper and does not contain all the details. To do variational inference, we condition our model parameters on their own hyper-parameters. Here we expand it into:

$$\begin{aligned} q(\beta,\Omega) =& (\prod_{k=1}^{K} q(\beta_k|\lambda_k)q(v_k|\omega_k)) \\ & (\prod_{l=1}^{L} q(g_l|p_l)(\prod_{i=1}^{I} q(\epsilon_{l_i}|q_{l_i})q(h_{l_i}|\varepsilon_{l_i}))) \\ & (\prod_{d=1}^{D} q(e_d|\mu_d)\prod_{j=1}^{J} q(c_{dj}|\zeta_{dj})q(\pi_{dj}|\alpha_{dj})\prod_{n=1}^{N} q(z_{dn}|\phi_{dn})) \end{aligned} \tag{11}$$

This is the complete $variational\ distribution$. From this, we can deduce the $complete\ conditional$ for every parameter. A complete conditional is the distribution of a parameter given all the other parameters. We also assume the conditional distribution of parameters on their hyper-parameters are also from the same exponential families. So $q(z|\phi)$, $q(c|\zeta)$ and $q(h|\varepsilon)$ are Multinomial distributions. $q(\pi|\alpha^1,\alpha^2)$, $q(\epsilon|q^1,q^2)$, $q(g|p^1,p^2)$ and $q(v|\omega^1,\omega^2)$ are Beta distributions. Finally, $q(\beta|\lambda)$ is Dirichlet distribution.

We abuse the notation a bit here. We convert our denotations into vector indicators. For instance, we treat w as a vector of size($\mathbb{S}$). So if the nth agent state in dth data segment is v, it can be represented by $w_{d_n}^v = 1$. $z_{d_n}^j = 1$ means the nth agent state in the dth data segment is classified into the jth group in this segment. Similarly, $c_{d_j}^{l_i} = 1$ means the jth group in the dth data segment is assigned to the ith component in cluster l. Finally, $h_{l_i}^k = 1$ means the ith component in the lth cluster is assigned to the kth pattern. So the complete conditionals for Multinomial nodes are:

$$\begin{aligned} &P(z_{d_n}^j = 1|\pi_d, w_{d_n}, c_d, e_d, h, \beta) \\ &\quad \propto exp\{log\sigma_j(\pi_d) + \sum_{l=1}^{L} e_d^l \sum_{i=1}^{I} c_{d_j}^{l_i} \sum_{k=1}^{K} h_{l_i}^k log\beta_{k,w_{d_n}}\} \end{aligned} \tag{12}$$

$$\begin{aligned} &P(c_{d_j}^{l_i} = 1|w_d, z_d, e_d, h, \epsilon, \beta) \\ &\quad \propto exp\{log\sigma_i(\epsilon_l) + e_d^l \sum_{n=1}^{N} z_{d_n}^j \sum_{k=1}^{K} h_{l_i}^k log\beta_{k,w_{d_n}}\} \end{aligned} \tag{13}$$

$$\begin{aligned} &P(e_d^l = 1|g, w_d, z_d, c_d, h, \beta) \\ &\quad \propto exp\{log\sigma_l(g) + \sum_{i=1}^{I}\sum_{j=1}^{J} c_{d_j}^{l_i} \sum_{n=1}^{N} z_{d_n}^j \sum_{k=1}^{K} h_{l_i}^k log\beta_{k,w_{d_n}}\} \end{aligned} \tag{14}$$

$$\begin{aligned} &P(h_{l_i}^k = 1|v, w, z, e, c, \beta) \\ &\quad \propto exp\{log\sigma_k(v) + \sum_{d=1}^{D} e_d^l \sum_{j=1}^{J} c_{d_j}^{l_i} \sum_{n=1}^{N} z_{d_n}^j log\beta_{k,w_{d_n}}\} \end{aligned} \tag{15}$$

Aside from the Multinomial nodes, we also have Beta nodes:

$$P(v_k|h,w) = Beta(1+\sum_{l=1}^{L}\sum_{i=1}^{I} h_{l_i}^k, \omega + \sum_{l=1}^{L}\sum_{i=1}^{I}\sum_{m>k} h_{l_i}^m) \tag{16}$$

$$P(g_l|b,e) = Beta(1+\sum_{d=1}^{D} e_d^l, b + \sum_{d=1}^{D}\sum_{m>l} e_d^l) \tag{17}$$

$$P(\epsilon_{l_i}|a,c) = Beta(1+\sum_{d=1}^{D}\sum_{j=1}^{J} c_{d_j}^{l_i}, a+\sum_{d=1}^{D}\sum_{j=1}^{J}\sum_{m>i} c_{d_j}^{l_m}) \quad (18)$$

$$P(\pi_{d_j}|\alpha,z_d) = Beta(1+\sum_{n=1}^{N} z_{d_n}^{j}, \sum_{n=1}^{N}\sum_{m>j} z_{d_n}^{m}) \quad (19)$$

Finally, the path patterns are Dirichlet distributions:

$$P(\beta_k|w,z,c,e,h,\eta) = Dirichlet(\eta+\sum_{d=1}^{D}\sum_{l=1}^{L} e_d^l \sum_{i=1}^{I} h_{l_i}^k \sum_{j=1}^{J} c_{d_j}^{l_i} \sum_{n=1}^{N} z_{d_n}^{j} w_{d_n}) \quad (20)$$

Given the complete conditionals, now we can compute the hyper-parameters. We first give the distributions of hyper-parameters of the Multinomial distributions:

$$\phi_{d_n}^{j} = \mathbb{E}[z_{d_n}^{j}] \propto exp\{log\sigma_j(\pi_d)+ \sum_{l=1}^{L}\mu_d^l \sum_{i=1}^{I}\zeta_{d_j}^{l_i}\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\mathbb{E}[log\beta_{k,w_{d_n}}]\} \quad (21)$$

$$\zeta_{d_j}^{l_i} = \mathbb{E}[c_{d_j}^{l_i}] = \propto exp\{log\sigma_i(\epsilon_l) + \mu_d^l \sum_{n=1}^{N}\phi_{d_n}^{j}\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\mathbb{E}[log\beta_{k,w_{d_n}}]\} \quad (22)$$

$$\mu_d^l = \mathbb{E}[e_d^l] \propto exp\{log\sigma_l(g)+ \sum_{i=1}^{I}\sum_{j=1}^{J}\zeta_{d_j}^{l_i}\sum_{n=1}^{N}\phi_{d_n}^{j}\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\mathbb{E}[log\beta_{k,w_{d_n}}]\} \quad (23)$$

$$\varepsilon_{l_i}^{k} = \mathbb{E}[h_{l_i}^{k}] \propto exp\{log\sigma_k(v)+ \sum_{d=1}^{D}\mu_d^l \sum_{j=1}^{J}\zeta_{d_j}^{l_i}\sum_{n=1}^{N}\phi_{d_n}^{j}\mathbb{E}[log\beta_{k,w_{d_n}}]\} \quad (24)$$

(25)

We also give the distributions of hyper-parameters of the Beta distributions:

$$o_k = (1+\sum_{l=1}^{L}\sum_{i=1}^{I}\varepsilon_{l_i}^{k}, \omega+\sum_{l=1}^{L}\sum_{i=1}^{I}\sum_{m>k}\varepsilon_{l_i}^{m}) \quad (26)$$

$$p_l = (1+\sum_{d=1}^{D}\mu_d^l, b+\sum_{d=1}^{D}\sum_{m>l}\mu_d^l) \quad (27)$$

$$q_{l_i} = (1+\sum_{d=1}^{D}\sum_{j=1}^{J}\zeta_{d_j}^{l_i}, a+\sum_{d=1}^{D}\sum_{j=1}^{J}\sum_{m>i}\zeta_{d_j}^{l_m}) \quad (28)$$

$$\gamma_{d_j} = (1+\sum_{n=1}^{N}\phi_{d_n}^{j}, \alpha+\sum_{n=1}^{N}\sum_{m>j}\phi_{d_n}^{m}) \quad (29)$$

Finally, for the sake of completeness, we give the equations to calculate $\mathbb{E}[log\sigma_i(v)]$ and $\mathbb{E}[log\beta]$:

$$\begin{aligned}\mathbb{E}[logv_k] &= \Psi(o_k^1)-\Psi(o_k^1+o_k^2)\\ \mathbb{E}[log(1-v_k)] &= \Psi(o_k^2)-\Psi(o_k^1+o_k^2)\\ \mathbb{E}[log\sigma_k(v)] &= \mathbb{E}[logv_k]+\sum_{l=1}^{k-1}\mathbb{E}[log(1-v_k)] \quad (30)\\ \mathbb{E}[log\beta_{kv}] &= \Psi(\lambda_{kv})-\Psi(\sum_{v\prime}\lambda_{kv\prime}) \quad (31)\end{aligned}$$

where Ψ is digamma function.

Algorithm 3: initLocal

Data: w_d
Result: ε, ζ_d, ϕ_d
for $l \in \{1,\ldots,L\}$ **do**
  for $i \in \{1,\ldots,I\}$ **do**
    $\varepsilon_{l_i}^{k} \propto exp\{\sum_{n=1}^{N}\mathbb{E}[log\beta_{k,w_{dn}}]\}, k \in \{1,\ldots,K\}$;
  end
end
for $l \in \{1,\ldots,L\}$ **do**
  $\mu_d^l \propto exp\{\sum_{i=1}^{I}\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\sum_{n=1}^{N}\mathbb{E}[log\beta_{k,w_{dn}}]\}$;
end
for $j \in \{1,\ldots,J\}$ **do**
  for $l \in \{1,\ldots,L\}$ **do**
    $\zeta_{d_j}^{l_i} \propto exp\{\mu_d^l\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\sum_{n=1}^{N}\mathbb{E}[log\beta_{k,w_{dn}}]\}$, $i \in \{1,\ldots,I\}$;
  end
end
for $n \in \{1,\ldots,N\}$ **do**
  $\phi_{d_n}^{j} \propto exp\{\sum_{l=1}^{L}\mu_d^l\sum_{i=1}^{I}\zeta_{d_j}^{l_i}\sum_{k=1}^{K}\varepsilon_{l_i}^{k}\mathbb{E}[log\beta_{k,w_{dn}}]\}$, $j \in \{1,\ldots,J\}$;
end

3 Additional Patterns

3.1 Bi-directional Flows

Here, we show some data segments of the simulations done for our Bi-directional flow example. They are shown in Figure 2.

3.2 Park

Trajectories and data segments of the park dataset is shown in Figure 3.

3.3 Train Station

3.3.1 Patterns learned by SVDHDP

Figure 4 shows some snapshots of the data segments of the train station dataset. Some additional patterns for the train station dataset shown in Figure 5.

3.3.2 Patterns learned by Gibbs Sampling

The top 32 patterns learned by Gibbs Sampling for the train station dataset shown in Figure 6 and Figure 7.

Algorithm 4: opLocal

Data: w_d, ε, ζ_d, ϕ_d, β, g
Result: ζ_d, ϕ_d, μ_d
while μ_d *not converged* **do**
  while γ_d, ζ_d, ϕ_d *not converged* **do**
    for $j \in \{1, \ldots, J\}$ **do**
      $\gamma^1_{d_j} = 1 + \sum_{n=1}^{N} \phi^j_{d_n}$;
      $\gamma^2_{d_j} = \alpha + \sum_{n=1}^{N} \sum_{m>j} \phi^m_{d_n}$;
      $\zeta^{l_i}_{d_j} \propto exp\{\mathbb{E}[log\sigma_l(g))] + \mathbb{E}[log\sigma_i(\epsilon_l)] + \sum_{n=1}^{N} \phi^j_{d_n} \sum_{k=1}^{K} \varepsilon^k_{l_i} \mathbb{E}[log\beta_{k,w_{d_n}}], l \in \{1, \ldots, J\}, i \in \{1, \ldots, I\}$;
    end
    for $n \in \{1, \ldots, N\}$ **do**
      $\phi^j_{d_n} \propto exp\{\mathbb{E}[log\sigma_j(\pi_d)] + \sum_{l=1}^{L} \sum_{i=1}^{I} \zeta^{l_i}_{d_j} \sum_{k=1}^{K} \varepsilon^k_{l_i} \mathbb{E}[log\beta_{k,w_{d_n}}]\}, j \in \{1, \ldots . J\}$;
    end
  end
  $\mu^l_d \propto exp\{\mathbb{E}[log\sigma_l(g)] + \sum_{i=1}^{I} \sum_{j=1}^{J} \zeta^{l_i}_{d_j} \sum_{n=1}^{N} \phi^j_{d_n} \sum_{k=1}^{K} \varepsilon^k_{l_i} \mathbb{E}[log\beta_{k,w_{d_n}}]\}, l \in \{1, \ldots, L\}$
end

4 Similarity

4.1 Park Simulation

Here we show, in Figure 8, some data segments of the four simulations we used in similarity computation in the park example.

4.2 Train Station Simulation

Here we show, in Figure 9, some data segments of the four simulations we used in similarity computation in the train station example. Learned patterns can be found in the main paper.

References

AMARI, S.-I. 1998. Natural Gradient Works Efficiently in Learning. *Neural Comp. 10*, 2, 251–276.

BISHOP, C. 2007. *Pattern Recognition and Machine Learning*. Springer, New York.

MACEACHERN, S. 1999. Dependent Nonparametric Processes. In *ASA Bayesian Stat. Sci.*

ROBBINS, H., AND MONRO, S. 1951. A Stochastic Approximation Method. *Ann. Math. Statist. 22*, 3, 400–407.

SETHURAMAN, J. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica 4*, 639–650.

Algorithm 5: updateCluster

Data: w_d, ε_d, ζ_d, ϕ_d, v, β
Result: ε, p, q
Set initial step size $\rho^{t'}_l$, set initial $t' = 0$;
while *p not converged* **do**
  Set initial step size $\rho^{t'_o}_i$, set initial $t'_o = 0$;
  while *q, ε not converged* **do**
    $[\zeta_d, \phi_d]$ = opLocal(w_d, ε_d, ζ_d, ϕ_d, p, q)(Algorithm 4);
    for $l \in \{1, \ldots, L\}$ **do**
      for $i \in \{1, \ldots, I\}$ **do**
        $\hat{q}^1_{l_i} = 1 + \mathrm{D}\sum_{j=1}^{J} \zeta^{l_i}_{d_j}$;
        $\hat{q}^2_{l_i} = \mathrm{a} + \mathrm{D}\sum_{j=1}^{J} \sum_{m>i} \zeta^{l_m}_{d_j}$;
        $\hat{\varepsilon}^k_{l_i} \propto exp\{\mathbb{E}[log\sigma_k(v)] + D\mu^l_d \sum_{j=1}^{J} \zeta^{l_i}_{d_j} \sum_{n=1}^{N} \phi^j_{d_n} \mathbb{E}[log\beta_{k,w_{d_n}}]\}, k \in \{1, \ldots, K\}$;
      end
    end
    $t'_o = t'_o + 1$;
    update $\rho^{t'_o}_l$ with t'_o (Equation 9). **for** $l \in \{1, \ldots, L\}$ **do**
      for $i \in \{1, \ldots, I\}$ **do**
        $q^{(1,t')}_{l_i} = (1 - \rho^{t'}_l) q^{(1,t'-1)}_{l_i} + \rho^{t'_o}_l \hat{q}^1_{l_i}$;
        $q^{(2,t')}_{l_i} = (1 - \rho^{t'}_l) q^{(2,t'-1)}_{l_i} + \rho^{t'_o}_l \hat{q}^2_{l_i}$;
        $\varepsilon^{(k,t')}_{l_i} = (1 - \rho^{t'}_l) \varepsilon^{(k,t'-1)}_{l_i} + \rho^{t'_o}_l \hat{\varepsilon}^k_{l_i}$, $k \in \{1, \ldots, K\}$;
      end
    end
  end
  $t' = t' + 1$;
  update $\rho^{t'}_l$ with t' (Equation 9). **for** $l \in \{1, \ldots, L\}$ **do**
    $\hat{p}^1_l = 1 + \mathrm{D}\sum_{i=1}^{I} \mathbb{E}(\epsilon_{l_i}) \sum_{j=1}^{J} \zeta^{l_i}_{d_j}$;
    $\hat{p}^2_l = \mathrm{b} + \mathrm{D}\sum_{i=1}^{I} \mathbb{E}(\epsilon_{l_i}) \sum_{j=1}^{J} \sum_{m>l} \zeta^{m_i}_{d_j}$;
  end
  for $l \in \{1, \ldots, L\}$ **do**
    $p^{(1,t')}_l = (1 - \rho^{t'}_l) p^{(1,t'-1)}_l + \rho^{t'}_l \hat{p}^1_l$;
    $p^{(2,t')}_l = (1 - \rho^{t'}_l) p^{(2,t'-1)}_l + \rho^{t'}_l \hat{p}^2_l$;
  end
end

Algorithm 6: updateGlobal

Data: w_d, η, ε, μ_d, ζ_d, ϕ_d, ρ_t, $\lambda^{(t-1)}$, $o^{1,(t-1)}$, $o^{2,(t-1)}$
Result: $\lambda^{(t)}$, $o^{1,(t)}$, $o^{2,(t)}$
for $k \in \{1, \ldots, K\}$ **do**
  $\hat{\lambda}_{kv} = \eta + \mathrm{D}\sum_{l=1}^{L} \mu^l_d \sum_{i=1}^{I} \varepsilon^k_{l_i} \sum_{j=1}^{J} \zeta^{l_i}_{d_j} \sum_{n=1}^{N} \phi^j_{d_n} w^v_{d_n}$;
  $\hat{o}^1_k = 1 + \sum_{l=1}^{L} \sum_{i=1}^{I} \varepsilon^k_{l_i}$;
  $\hat{o}^2_k = \omega + \sum_{l=1}^{L} \sum_{i=1}^{I} \sum_{m>k} \varepsilon^m_{l_i}$;
end
$\lambda^{(t)} = (1 - \rho_t)\lambda^{(t-1)} + \rho_t \hat{\lambda}$;
$o^{1,(t)} = (1 - \rho_t) o^{1,(t-1)} + \rho_t \hat{o^1}$;
$o^{2,(t)} = (1 - \rho_t) o^{(2,t-1)} + \rho_t \hat{o^2}$;
return $\lambda^{(t)}$, $o^{1,(t)}$, $o^{2,(t)}$

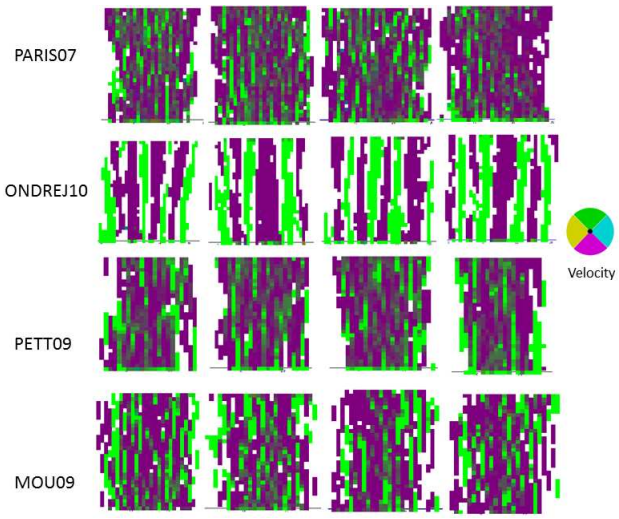

Figure 2: *Data segment samples from PARIS07, ONDREJ10, PETT09 and MOU09*

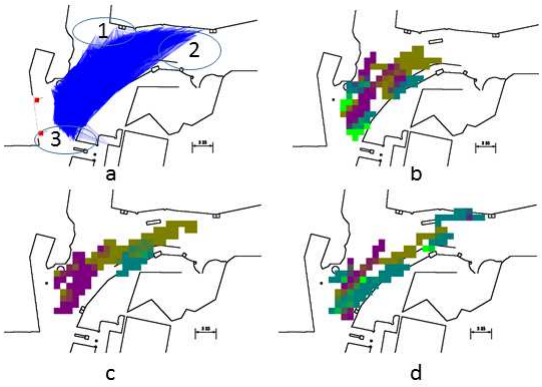

Figure 3: *a: All trajectories. The red dots are cameras. The blue circles are exts/entrances. b-d:data segments. All data segments span 5 seconds.*

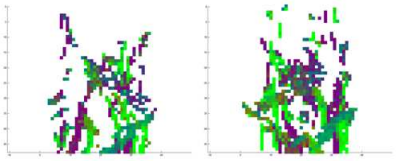

Figure 4: *Two data segments in train station dataset.*

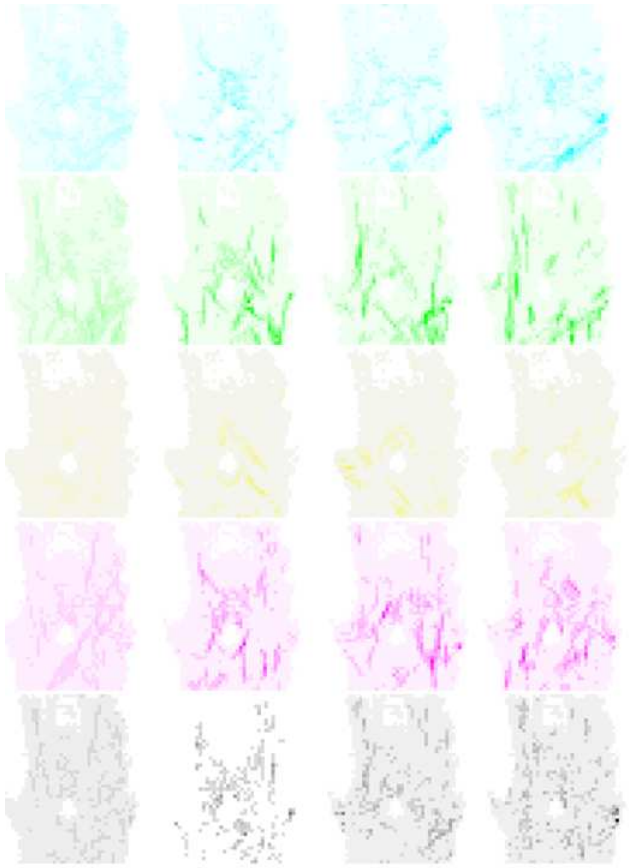

Figure 5: *Additional patterns learned by SVDHDP from train station dataset.*

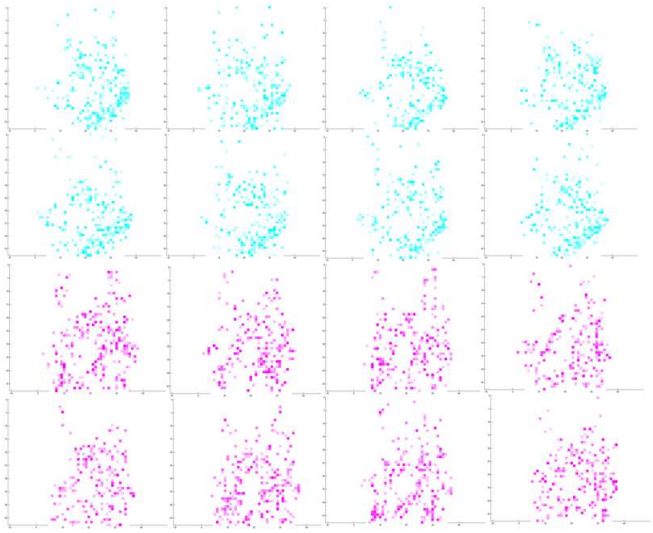

Figure 6: *Patterns learned Gibbs Sampling from train station dataset.*

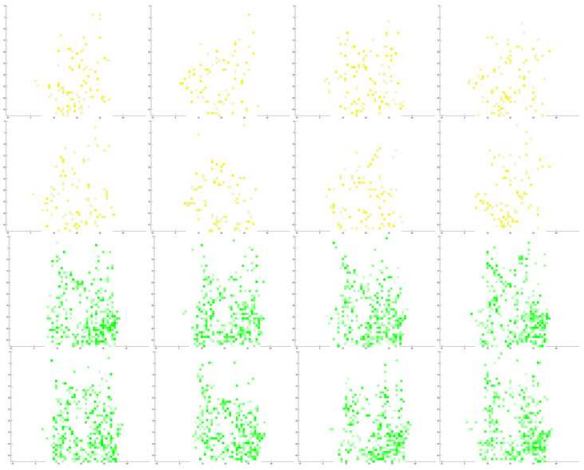

Figure 7: *Patterns learned Gibbs Sampling from train station dataset.*

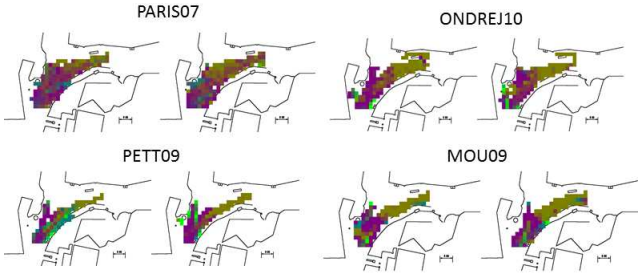

Figure 8: *Data segment samples for park simulation from PARIS07, ONDREJ10, PETT09 and MOU09*

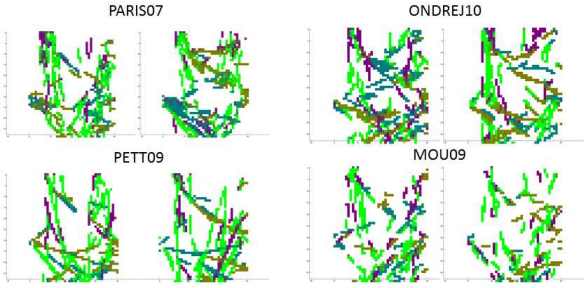

Figure 9: *Data segment samples for train station simulation from PARIS07, ONDREJ10, PETT09 and MOU09*