Memoized Online Variational Inference for Dirichlet Process Mixture Models

Michael C. Hughes and Erik B. Sudderth
Department of Computer Science, Brown University, Providence, RI 02912
mhughes@cs.brown.edu, sudderth@cs.brown.edu

Abstract

Variational inference algorithms provide the most effective framework for large-scale training of Bayesian nonparametric models. Stochastic online approaches are promising, but are sensitive to the chosen learning rate and often converge to poor local optima. We present a new algorithm, *memoized online variational inference*, which scales to very large (yet finite) datasets while avoiding the complexities of stochastic gradient. Our algorithm maintains finite-dimensional sufficient statistics from batches of the full dataset, requiring some additional memory but still scaling to millions of examples. Exploiting nested families of variational bounds for infinite nonparametric models, we develop principled birth and merge moves allowing non-local optimization. Births adaptively add components to the model to escape local optima, while merges remove redundancy and improve speed. Using Dirichlet process mixture models for image clustering and denoising, we demonstrate major improvements in robustness and accuracy.

1 Introduction

Bayesian nonparametric methods provide a flexible framework for unsupervised modeling of structured data like text documents, time series, and images. They are especially promising for large datasets, as their nonparametric priors should allow complexity to grow smoothly as more data is seen. Unfortunately, contemporary inference algorithms do not live up to this promise, scaling poorly and yielding solutions that represent poor local optima of the true posterior. In this paper, we propose new scalable algorithms capable of escaping local optima. Our focus is on clustering data via the Dirichlet process (DP) mixture model, but our methods are much more widely applicable.

Stochastic online variational inference is a promising general-purpose approach to Bayesian nonparametric learning from streaming data [1]. While individual steps of stochastic optimization algorithms are by design scalable, they are extremely vulnerable to local optima for non-convex unsupervised learning problems, frequently yielding poor solutions (see Fig. 2). While taking the best of multiple runs is possible, this is unreliable, expensive, and ineffective in more complex structured models. Furthermore, the noisy gradient step size (or learning rate) requires external parameters which must be fine-tuned for best performance, often requiring an expensive validation procedure. Recent work has proposed methods for automatically adapting learning rates [2], but these algorithms' progress on the overall variational objective remains local and non-monotonic.

In this paper, we present an alternative algorithm, *memoized online variational inference*, which avoids noisy gradient steps and learning rates altogether. Our method is useful when all data may not fit in memory, but we can afford multiple full passes through the data by processing successive batches. The algorithm visits each batch in turn and updates a cached set of sufficient statistics which accurately reflect the *entire dataset*. This allows rapid and noise-free updates to global parameters at every step, quickly propagating information and speeding convergence. Our memoized approach is generally applicable in any case batch or stochastic online methods are useful, including topic models [1] and relational models [3], though we do not explore these here.

We further develop a principled framework for escaping local optima in the online setting, by integrating birth and merge moves within our algorithm's coordinate ascent steps. Most existing mean-field algorithms impose a restrictive fixed truncation in the number of components, which is hard to set *a priori* on big datasets: either it is too small and inexpressive, or too large and computationally inefficient. Our birth and merge moves, together with a *nested* variational approximation to the posterior, enable adaptive creation and pruning of clusters on-the-fly. Because these moves are validated by an exactly tracked global variational objective, we avoid potential instabilities of stochastic online split-merge proposals [4]. The structure of our moves is very different from split-merge MCMC methods [5, 6]; applications of these algorithms have been limited to hundreds of data points, while our experiments show scaling of memoized split-merge proposals to millions of examples.

We review the Dirichlet process mixture model and variational inference in Sec. 2, outline our novel memoized algorithm in Sec. 3, and evaluate on clustering and denoising applications in Sec. 4.

2 Variational inference for Dirichlet process mixture models

The *Dirichlet process* (DP) provides a nonparametric prior for partitioning exchangeable datasets into discrete clusters [7]. An instantiation G of a DP is an infinite collection of atoms, each of which represents one mixture component. Component k has mixture weight w_k sampled as follows:

$$G \sim \text{DP}(\alpha_0 H), \quad G \triangleq \sum_{k=1}^{\infty} w_k \delta_{\phi_k}, \quad v_k \sim \text{Beta}(1, \alpha_0), \quad w_k = v_k \prod_{\ell=1}^{k-1} (1 - v_\ell). \tag{1}$$

This *stick-breaking* process provides mixture weights and parameters. Each data item n chooses an assignment $z_n \sim \text{Cat}(w)$, and then draws observations $x_n \sim F(\phi_{z_n})$. The data-generating parameter ϕ_k is drawn from a base measure H with natural parameters λ_0. We assume both H and F belong to exponential families with log-normalizers a and sufficient statistics t:

$$p(\phi_k \mid \lambda_0) = \exp\left\{\lambda_0^T t_0(\phi_k) - a_0(\lambda_0)\right\}, \qquad p(x_n \mid \phi_k) = \exp\left\{\phi_k^T t(x_n) - a(\phi_k)\right\}. \tag{2}$$

For simplicity, we assume unit reference measures. The goal of inference is to recover stick-breaking proportions v_k and data-generating parameters ϕ_k for each global mixture component k, as well as discrete cluster assignments $z = \{z_n\}_{n=1}^N$ for each observation. The joint distribution is

$$p(\mathbf{x}, \mathbf{z}, \phi, v) = \prod_{n=1}^{N} F(x_n \mid \phi_{z_n}) \text{Cat}(z_n \mid w(v)) \prod_{k=1}^{\infty} \text{Beta}(v_k \mid 1, \alpha_0) H(\phi_k \mid \lambda_0) \tag{3}$$

While our algorithms are directly applicable to any DP mixture of exponential families, our experiments focus on D-dimensional real-valued data x_n, for which we take F to be Gaussian. For some data, we consider full-mean, full-covariance analysis (where H is normal-Wishart), while other applications consider zero-mean, full-covariance analysis (where H is Wishart).

2.1 Mean-field variational inference for DP mixture models

To approximate the full (but intractable) posterior over variables z, v, ϕ, we consider a fully-factorized variational distribution q, with individual factors from appropriate exponential families:[1]

$$q(\mathbf{z}, v, \phi) = \prod_{n=1}^{N} q(z_n | \hat{r}_n) \prod_{k=1}^{K} q(v_k | \hat{\alpha}_1, \hat{\alpha}_0) q(\phi_k | \hat{\lambda}_k), \tag{4}$$

$$q(z_n) = \text{Cat}(z_n \mid \hat{r}_{n1}, \dots \hat{r}_{nK}), \quad q(v_k) = \text{Beta}(v_k \mid \hat{\alpha}_{k1}, \hat{\alpha}_{k0}), \quad q(\phi_k) = H(\phi_k \mid \hat{\lambda}_k). \tag{5}$$

To tractably handle the infinite set of components available under the DP prior, we truncate the discrete assignment factor to enforce $q(z_n = k) = 0$ for $k > K$. This forces all data to be explained by only the first K components, inducing *conditional independence* between observed data and any global parameters v_k, ϕ_k with index $k > K$. Inference may thus focus exclusively on a finite set of K components, while reasonably approximating the true infinite posterior for large K.

[1] To ease notation, we mark variables with hats to distinguish parameters $\hat{\theta}$ of variational factors q from parameters θ of the generative model p. In this way, θ_k and $\hat{\theta}_k$ always have equal dimension.

Crucially, our truncation is *nested*: any learned q with truncation K can be represented exactly under truncation $K+1$ by setting the final component to have zero mass. This truncation, previously advocated by [8, 4], has considerable advantages over non-nested direct truncation of the stick-breaking process [7], which places artifically large mass on the final component. It is more efficient and broadly applicable than an alternative trunction which sets the stick-breaking "tail" to its prior [9].

Variational algorithms optimize the parameters of q to minimize the KL divergence from the true, intractable posterior [7]. The optimal q maximizes the *evidence lower bound* (ELBO) objective $\mathcal{L}$:

$$\log p(\mathbf{x} \mid \alpha_0, \lambda_0) \geq \mathcal{L}(q) \triangleq \mathbb{E}_q\Big[\log p(\mathbf{x}, v, \mathbf{z}, \phi \mid \alpha_0, \lambda_0) - \log q(v, \mathbf{z}, \phi)\Big] \tag{6}$$

For DP mixtures of exponential family distributions, $\mathcal{L}(q)$ has a simple form. For each component k, we store its expected mass $\hat{N}_k$ and expected sufficient statistic $s_k(\mathbf{x})$. All but one term in $\mathcal{L}(q)$ can then be written using only these summaries and expectations of the global parameters v, ϕ:

$$\hat{N}_k \triangleq \mathbb{E}_q\Big[\sum_{n=1}^{N} z_{nk}\Big] = \sum_{n=1}^{N} \hat{r}_{nk}, \qquad s_k(\mathbf{x}) \triangleq \mathbb{E}_q\Big[\sum_{n=1}^{N} z_{nk} t(x_n)\Big] = \sum_{n=1}^{N} \hat{r}_{nk} t(x_n), \tag{7}$$

$$\begin{aligned}\mathcal{L}(q) = \sum_{k=1}^{K} \Bigg(& \mathbb{E}_q[\phi_k]^T s_k(\mathbf{x}) - \hat{N}_k \mathbb{E}_q[a(\phi_k)] + \hat{N}_k \mathbb{E}_q[\log w_k(v)] - \sum_{n=1}^{N} \hat{r}_{nk} \log \hat{r}_{nk} \\ & + \mathbb{E}_q\left[\log \frac{\text{Beta}(v_k \mid 1, \alpha_0)}{q(v_k \mid \hat{\alpha}_{k1}, \hat{\alpha}_{k0})}\right] + \mathbb{E}_q\left[\log \frac{H(\phi_k \mid \lambda_0)}{q(\phi_k \mid \hat{\lambda}_k)}\right]\Bigg)\end{aligned} \tag{8}$$

Excluding the entropy term $-\sum \hat{r}_{nk} \log \hat{r}_{nk}$ which we discuss later, this bound is a simple linear function of the summaries $\hat{N}_k, s_k(\mathbf{x})$. Given precomputed entropies and summaries, evaluation of $\mathcal{L}(q)$ can be done in time *independent* of the data size N. We next review variational algorithms for optimizing q via coordinate ascent, iteratively updating individual factors of q. We describe algorithms in terms of two updates [1]: *global* parameters (stick-breaking proportions v_k and data-generating parameters ϕ_k), and *local* parameters (assignments of data to components z_n).

2.2 Full-dataset variational inference

Standard full-dataset variational inference [7] updates local factors $q(z_n \mid \hat{r}_n)$ for *all* observations $n = 1, \ldots, N$ by visiting each item n and computing the fraction $\hat{r}_{nk}$ explained by component k:

$$\tilde{r}_{nk} = \exp\Big(\mathbb{E}_q[\log w_k(v)] + \mathbb{E}_q[\log p(x_n \mid \phi_k)]\Big), \quad \hat{r}_{nk} = \frac{\tilde{r}_{nk}}{\sum_{\ell=1}^{K} \tilde{r}_{n\ell}}. \tag{9}$$

Next, we update global factors $q(v_k|\hat{\alpha}_{k1}, \hat{\alpha}_{k0}), q(\phi_k|\hat{\lambda}_k)$ for each component k. After computing summary statistics $\hat{N}_k, s_k(\mathbf{x})$ given the new $\hat{r}_{nk}$ via Eq. (7), the update equations become

$$\hat{\alpha}_{k1} = \alpha_1 + \hat{N}_k, \qquad \hat{\alpha}_{k0} = \alpha_0 + \sum_{\ell=k+1}^{K} \hat{N}_\ell, \qquad \hat{\lambda}_k = \lambda_0 + s_k(\mathbf{x}). \tag{10}$$

While simple and guaranteed to converge, this approach scales poorly to big datasets. Because global parameters are updated only after a full pass through the data, information propagates slowly.

2.3 Stochastic online variational inference

Stochastic online (SO) variational inference scales to huge datasets [1]. Instead of analyzing all data at once, SO processes only a subset ("batch") $\mathcal{B}_t$ at each iteration t. These subsets are assumed sampled uniformly at random from a larger (but *fixed* size N) corpus. Given a batch, SO first updates local factors $q(z_n)$ for $n \in \mathcal{B}_t$ via Eq. (9). It then updates global factors via a *noisy gradient* step, using sufficient statistics of $q(z_n)$ from only the current batch. These steps optimize a noisy function, which in expectation (with respect to batch sampling) converges to the true objective (6).

Natural gradient steps are computationally tractable for exponential family models, involving nearly the same computations as the full-dataset updates [1]. For example, to update the variational parameter $\hat{\lambda}_k$ from (5) at iteration t, we first compute the global update given only data in the current batch,

amplified to be at full-dataset scale: $\hat{\lambda}_k^* = \lambda_0 + \frac{N}{|\mathcal{B}_t|} s_k(\mathcal{B}_t)$. Then, we interpolate between this and the previous global parameters to arrive at the final result: $\hat{\lambda}_k^{(t)} \leftarrow \rho_t \hat{\lambda}_k^* + (1-\rho_t)\hat{\lambda}_k^{(t-1)}$. The learning rate ρ_t controls how "forgetful" the algorithm is of previous values; if it decays at appropriate rates, stochastic inference provably converges to a *local* optimum of the global objective $\mathcal{L}(q)$ [1].

This online approach has clear computational advantages and can sometimes yield higher quality solutions than the full-data algorithm, since it conveys information between local and global parameters more frequently. However, performance is extremely sensitive to the learning rate decay schedule and choice of batch size, as we demonstrate in later experiments.

3 Memoized online variational inference

Generalizing previous incremental variants of the *expectation maximization* (EM) algorithm [10], we now develop our *memoized online variational inference* algorithm. We divide the data into B fixed batches $\{\mathcal{B}_b\}_{b=1}^B$. For each batch, we maintain *memoized* sufficient statistics $S_k^b = [\hat{N}_k(\mathcal{B}_b), s_k(\mathcal{B}_b)]$ for each component k. We also track the full-dataset statistics $S_k^0 = [\hat{N}_k, s_k(\mathbf{x})]$. These compact summary statistics allow guarantees of correct full-dataset analysis while processing only one small batch at a time. Our approach hinges on the fact that these sufficient statistics are *additive*: summaries of an entire dataset can be written exactly as the addition of summaries of distinct batches. Note that our memoization of deterministic analyses of batches of data is distinct from the stochastic memoization, or "lazy" instantiation, of random variables in some Monte Carlo methods [11, 12].

Memoized inference proceeds by visiting (in random order) each distinct batch once in a full pass through the data, incrementally updating the local and global parameters related to that batch b. First, we update local parameters for the current batch ($q(z_n \mid \hat{r}_n)$ for $n \in \mathcal{B}_b$) via Eq. (9). Next, we update cached global sufficient statistics for each component: we subtract the old (cached) summary of batch b, compute a new batch-level summary, and add the result to the full-dataset summary:

$$S_k^0 \leftarrow S_k^0 - S_k^b, \quad S_k^b \leftarrow \Big[\sum_{n \in \mathcal{B}_b} \hat{r}_{nk}, \sum_{n \in \mathcal{B}_b} \hat{r}_{nk} t(x_n) \Big], \quad S_k^0 \leftarrow S_k^0 + S_k^b. \tag{11}$$

Finally, given the new full-dataset summary S_k^0, we update global parameters exactly as in Eq. (10). Unlike stochastic online algorithms, memoized inference is guaranteed to improve the full-dataset ELBO at every step. Correctness follows immediately from the arguments in [10]. By construction, each local or global step will result in a new q that strictly increases the objective $\mathcal{L}(q)$.

In the limit where $B = 1$, memoized inference reduces to standard full-dataset updates. However, given many batches it is far more scalable, while maintaining all guarantees of batch inference. Furthermore, it generally converges faster than the full-dataset algorithm due to frequently interleaving global and local updates. Provided we can store memoized sufficient statistics for each batch (*not* each observation), memoized inference has the same computational complexity as stochastic methods while avoiding noise and sensitivity to learning rates. Recent analysis of convex optimization algorithms [13] demonstrated theoretical and practical advantages for methods that use cached full-dataset summaries to update parameters, as we do, instead of stochastic current-batch-only updates.

This memoized algorithm can compute the full-dataset objective $\mathcal{L}(q)$ exactly at any point (after visiting all items once). To do so efficiently, we need to compute and store the assignment entropy $H_k^b = -\sum_{n \in \mathcal{B}_b} r_{nk} \log r_{nk}$ after visiting each batch b. We also need to track the full-data entropy $H_k^0 = \sum_{b=1}^B H_k^b$, which is additive just like the sufficient statistics and incrementally updated after each batch. Given both H_k^0 and S_k^0, evaluation of the full-dataset ELBO in Eq. (8) is exact and rapid.

3.1 Birth moves to escape local optima

We now propose additional *birth* moves that, when interleaved with conventional coordinate ascent parameter updates, can add useful new components to the model and escape local optima. Previous methods for changing variational truncations create just one extra component via a "split" move that is highly-specialized to particular likelihoods. Wang and Blei [15] explore truncation levels via a local collapsed Gibbs sampler, but samplers are slow to make large changes. In contrast, our births add *many* components at once and apply to *any* exponential family mixture model.

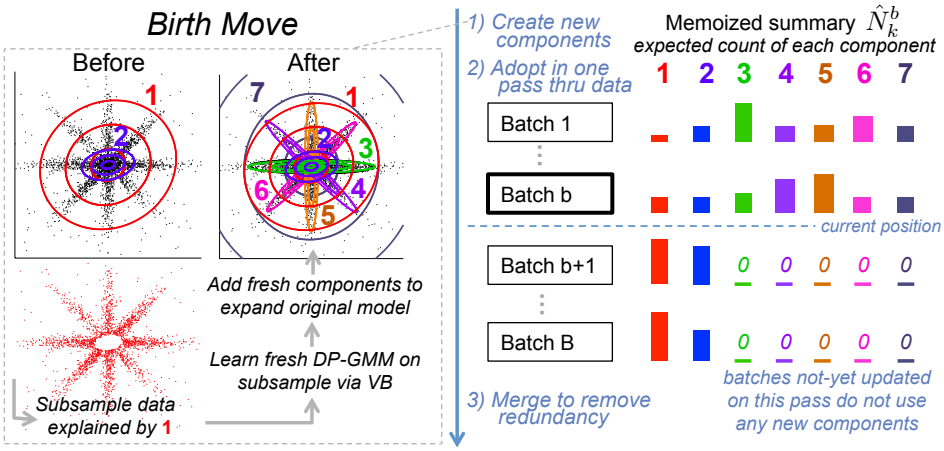

Figure 1: One pass through a toy dataset for memoized learning with birth and merge moves (MO-BM), showing creation (left) and adoption (right) of new components. *Left:* Scatter plot of 2D observed data, and a subsample targeted via the first mixture component. Elliptical contours show component covariance matrices. *Right:* Bar plots of memoized counts $\hat{N}_k^b$ for each batch. *Not shown:* Memoized sufficient statistics s_k^b.

Creating new components in the online setting is challenging. Each small batch may not have enough examples of a missing component to inspire a good proposal, even if that component is well-supported by the full dataset. We thus advocate birth moves that happen in three phases (collection, creation, and adoption) over two passes of the data. The first pass collects a targeted data sample more likely to yield informative proposals than a small, predefined batch. The second pass, shown in Fig. 1, creates new components and then updates every batch with the expanded model. Successive births are interleaved; at each pass there are proposals both active and in preparation. We sketch out each step of the algorithm below. For complete details, see the supplement.

Collection During pass 1, we collect a *targeted* subsample $\mathbf{x}'$ of the data, of size at most $N' = 10^4$. This subsample targets a single component k'. When visiting each batch, we copy data x_n into $\mathbf{x}'$ if $\hat{r}_{nk'} > \tau$ (we set $\tau = 0.1$). This threshold test ensures the subsample contains related data, but also promotes diversity by considering data explained partially by other components $k \neq k'$. Targeted samples vary from iteration to iteration because batches are visited in distinct, random orders.

Creation Before pass 2, we *create* new components by fitting a DP mixture model with K' (we take $K' = 10$) components to $\mathbf{x}'$, running variational inference for a limited budget of iterations. Taking advantage of our nested truncation, we expand our current model to include all $K + K'$ components, as shown in Fig. 1. Unlike previous work [9, 4], we do not immediately assess the change in ELBO produced by these new components, and always accept them. We rely on subsequent merge moves (Sec. 3.2) to remove unneeded components.

Adoption During pass 2, we visit each batch and perform local and global parameter updates for the expanded $(K + K')$-component mixture. These updates use expanded global summaries S^0 that include summaries S' from the targeted analysis of $\mathbf{x}'$. This results in *two* interpretations of the subset $\mathbf{x}'$: assignment to original components (mostly k') and assignment to brand-new components. If the new components are favored, they will gain mass via new assignments made at each batch. After the pass, we subtract away S' to yield both S^0 and global parameters exactly consistent with the data $\mathbf{x}$. Any nearly-empty new component will likely be pruned away by later merges.

By adding *many* components at once, our birth move allows rapid escape from poor local optima. Alone, births may sometimes cause a slight ELBO decrease by adding unnecessary components. However, in practice merge moves reliably reject poor births and restore original configurations. In Sec. 4, births are so effective that runs started at $K = 1$ recover necessary components on-the-fly.

3.2 Merge moves that optimize the full data objective

The computational cost of inference grows with the number of components K. To keep K small, we develop merge moves that replace two components with a single merged one. Merge moves were first explored for batch variational methods [16, 14]. For hierarchical DP topic models, stochastic

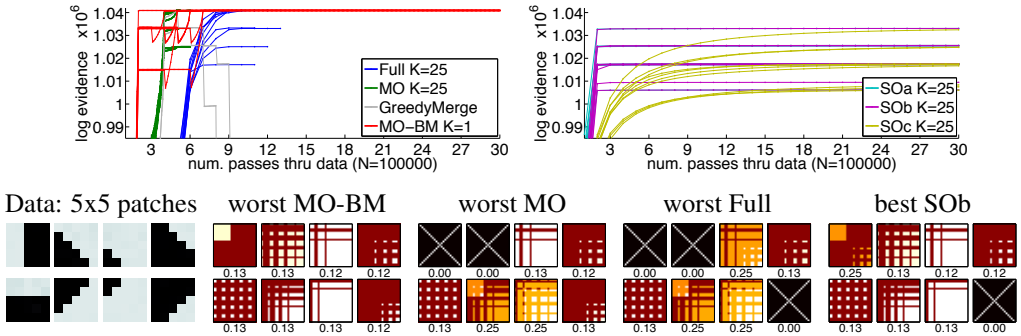

Figure 2: Comparison of full-data, stochastic (SO), and memoized (MO) on toy data with $K = 8$ true components (Sec. 4.1). *Top:* Trace of ELBO during training across 10 runs. SO compared with learning rates a,b,c. *Bottom Left:* Example patch generated by each component. *Bottom:* Covariance matrix and weights w_k found by one run of each method, aligned to true components. "X": no comparable component found.

variational inference methods have been augmented to evaluate merge proposals based on noisy, single-batch estimates of the ELBO [4]. This can result in accepted merges that *decrease* the full-data objective (see Sec. 4.1 for an empirical illustration). In contrast, our algorithm accurately computes the full ELBO for each merge proposal, ensuring only useful merges are accepted.

Given two components k_a, k_b to merge, we form a candidate q' with $K-1$ components, where merged component k_m takes over all assignments to k_a, k_b: $\hat{r}_{nk_m} = \hat{r}_{nk_a} + \hat{r}_{nk_b}$. Instead of computing new assignments explicitly, additivity allows direct construction of merged global sufficient statistics: $S^0_{k_m} = S^0_{k_a} + S^0_{k_b}$. Merged global parameters follow from Eq. (10).

Our merge move has three steps: select components, form the candidate configuration q', and accept q' if the ELBO improves. Selecting k_a, k_b to merge at random is unlikely to yield an improved configuration. After choosing k_a at random, we select k_b using a ratio of marginal likelihoods M which compares the merged and separated configurations, easily computed with cached summaries:

$$p(k_b \mid k_a) \propto \frac{M(S_{k_a} + S_{k_b})}{M(S_{k_a})M(S_{k_b})}, \quad M(S_k) = \exp\Big(a_0(\lambda_0 + s_k(\mathbf{x}))\Big). \tag{12}$$

Our memoized approach allows *exact* evaluation of the full-data ELBO to compare the existing q to merge candidate q'. As shown in Eq. (8), evaluating $\mathcal{L}(q')$ is a linear function of merged sufficient statistics, except for the assignment entropy term: $H_{ab} = -\sum_{n=1}^{N}(\hat{r}_{nk_a} + \hat{r}_{nk_b})\log(\hat{r}_{nk_a} + \hat{r}_{nk_b})$. We compute this term in advance for *all* possible merge pairs. This requires storing one set of $K(K-1)/2$ scalars, one per candidate pair, for each batch. This modest precomputation allows rapid and exact merge moves, which improve model quality *and* speed-up post-merge iterations.

In one pass of the data, our algorithm performs a birth, memoized ascent steps for all batches, and several merges after the final batch. After a few passes, it recovers high-quality, compact structure.

4 Experimental results

We now compare algorithms for learning DP-Gaussian mixture models (DP-GMM), using our own implementations of full-dataset, stochastic online (SO), and memoized online (MO) inference, as well as our new birth-merge memoized algorithm (MO-BM). Code is available online. To examine SO's sensitivity to learning rate, we use a recommended [1] decay schedule $\rho_t = (t+d)^{-\kappa}$ with three diverse settings: a) $\kappa = 0.5, d = 10$, b) $\kappa = 0.5, d = 100$, and c)$\kappa = 0.9, d = 10$.

4.1 Toy data: How reliably do algorithms escape local optima?

We first study $N = 100000$ synthetic image patches generated by a zero-mean GMM with 8 equally-common components. Each one is defined by a 25×25 covariance matrix producing 5×5 patches with a strong edge. We investigate whether algorithms recover the true $K = 8$ structure. Each fixed-truncation method runs from 10 fixed random initializations with $K = 25$, while MO-BM starts at $K = 1$. Online methods traverse 100 batches (1000 examples per batch).

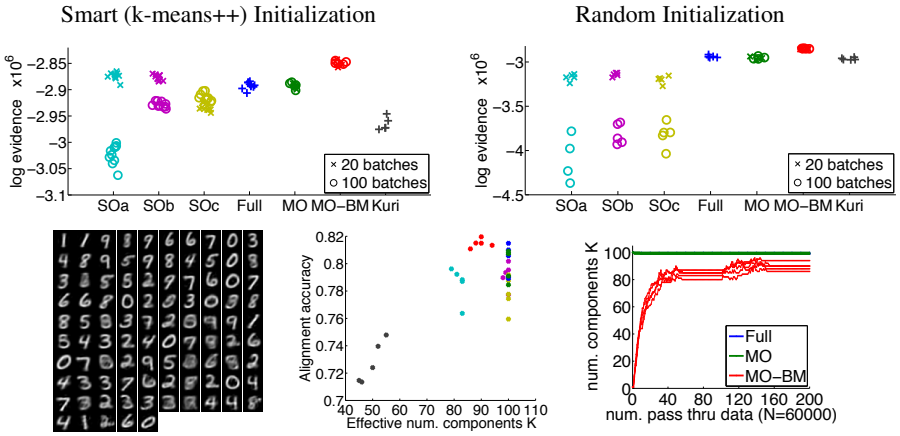

Figure 3: MNIST. *Top:* Comparison of final ELBO for multiple runs of each method, varying initialization and number of batches. Stochastic online (SO) compared at learning rates a,b,c. *Bottom left:* Visualization of cluster means for MO-BM's best run. *Bottom center:* Evaluation of cluster alignment to true digit label. *Bottom right:* Growth in truncation-level K as more data visited with MO-BM.

Fig. 2 traces the training-set ELBO as more data arrives for each algorithm and shows estimated covariance matrices for the top 8 components for select runs. Even the best runs of SO do not recover ideal structure. In contrast, all 10 runs of our birth-merge algorithm find all 8 components, despite initialization at $K = 1$. The ELBO trace plots show this method escaping local optima, with slight drops indicating addition of new components followed by rapid increases as these are adopted. They further suggest that our fixed-truncation memoized method competes favorably with full-data inference, often converging to similar or better solutions after fewer passes through the data.

The fact that our MO-BM algorithm only performs merges that improve the full-data ELBO is crucial. Fig. 2 shows trace plots of GreedyMerge, a memoized online variant that instead uses only the current-batch ELBO to assess a proposed merge, as done in [4]. Given small batches (1000 examples each), there is not always enough data to warrant many distinct 25×25 covariance components. Thus, this method favors merges that in fact remove vital structure. All 5 runs of this GreedyMerge algorithm ruinously accept merges that decrease the full objective, consistently collapsing down to just one component. Our memoized approach ensures merges are always globally beneficial.

4.2 MNIST digit clustering

We now compare algorithms for clustering $N = 60000$ MNIST images of handwritten digits 0-9. We preprocess as in [9], projecting each image down to $D = 50$ dimensions via PCA. Here, we also compare to Kurihara's public implementation of variational inference with split moves [9]. MO-BM and Kurihara start at $K = 1$, while other methods are given 10 runs from two $K = 100$ initialization routines: random and smart (based on k-means++ [17]). For online methods, we compare 20 and 100 batches, and three learning rates. All runs complete 200 passes through the full dataset.

The final ELBO values for every run of each method are shown in Fig. 3. SO's performance varies dramatically across initialization, learning rate, and number of batches. Under random initialization, SO reaches especially poor local optima (note lower y-axis scale). In contrast, our memoized approach consistently delivers solutions on par with full inference, with no apparent sensitivity to the number of batches. With births and merges enabled, MO-BM expands from $K = 1$ to over 80 components, finding better solutions than every smart $K = 100$ initialization. MO-BM even outperforms Kurihara's offline split algorithm, yielding 30-40 more components and higher ELBO values. Altogether, Fig. 3 exposes SO's extreme sensitivity, validates MO as a more reliable alternative, and shows that our birth-merge algorithm is more effective at avoiding local optima.

Fig. 3 also shows cluster means learned by the best MO-BM run, covering many styles of each digit. We further compute a hard segmentation of the data using the $q(z)$ from smart initialization runs. Each DP-GMM cluster is aligned to one digit by majority vote of its members. A plot of alignment accuracy in Fig. 3 shows our MO-BM consistently among the best, with SO lagging significantly.

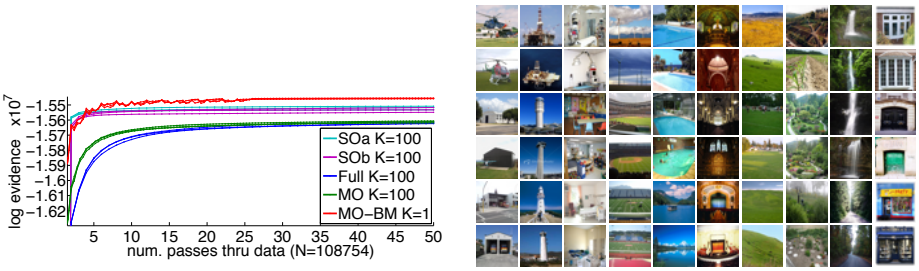

Figure 4: SUN-397 tiny images. *Left:* ELBO during training. *Right:* Visualization of 10 of 28 learned clusters for best MO-BM run. Each column shows two images from the top 3 categories aligned to one cluster.

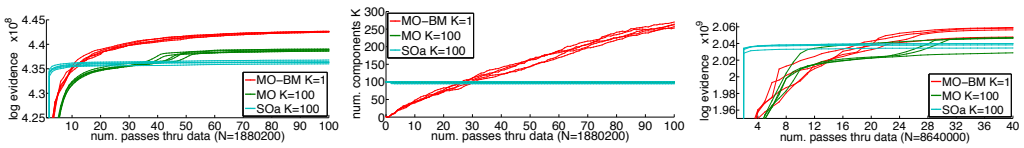

Figure 5: 8×8 image patches. *Left:* ELBO during training, $N = 1.88$ million. *Center:* Effective truncation-level K during training, $N = 1.88$ million. *Right:* ELBO during training, $N = 8.64$ million.

4.3 Tiny image clustering

We next learn a full-mean DP-GMM for tiny, 32×32 images from the SUN-397 scene categories dataset [18]. We preprocess all 108754 color images via PCA, projecting each example down to $D = 50$ dimensions. We start MO-BM at $K = 1$, while other methods have fixed $K = 100$. Fig. 4 plots the training ELBO as more data is seen. Our MO-BM runs surpass all other algorithms.

To verify quality, Fig. 4 shows images from the 3 most-related scene categories for each of several clusters found by MO-BM. For each learned cluster k, we rank all 397 categories to find those with the largest fraction of members assigned to k via $\hat{r}_{\cdot k}$. The result is quite sensible, with clusters for tall free-standing objects, swimming pools and lakes, doorways, and waterfalls.

4.4 Image patch modeling

Our last experiment applies a zero-mean, full-covariance DP-GMM to learn the covariance structures of natural image patches, inspired by [19, 20]. We compare online algorithms on $N = 1.88$ million 8×8 patches, a dense subsampling of all patches from 200 images of the Berkeley Segmentation dataset. Fig. 5 shows that our birth-merge memoized algorithm started at $K = 1$ can consistently add useful components and reach better solutions than alternatives. We also examined a much bigger dataset of $N = 8.64$ million patches, and still see advantages for our MO-BM.

Finally, we perform denoising on 30 heldout images, using code from [19]. Our best MO-BM run on the 1.88 million patch dataset achieves PSNR of 28.537 dB, within 0.05 dB of the PSNR achieved by [19]'s publicly-released GMM with $K = 200$ trained on a similar corpus. This performance is visually indistinguishable, highlighting the practical value of our new algorithm.

5 Conclusions

Our novel memoized online variational algorithm avoids noisiness and sensitivity inherent in stochastic methods. Our birth and merge moves successfully escape local optima. These innovations are applicable to common nonparametric models beyond the Dirichlet process.

Acknowledgments This research supported in part by ONR Award No. N00014-13-1-0644. M. Hughes supported in part by an NSF Graduate Research Fellowship under Grant No. DGE0228243.

References

[1] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, 14:1303–1347, 2013.

[2] R. Ranganath, C. Wang., D. Blei, and E. Xing. An adaptive learning rate for stochastic variational inference. In *ICML*, 2013.

[3] P. Gopalan, D. M. Mimno, S. Gerrish, M. J. Freedman, and D. M. Blei. Scalable inference of overlapping communities. In *NIPS*, 2012.

[4] M. Bryant and E. Sudderth. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *NIPS*, 2012.

[5] S. Jain and R.M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004.

[6] D. B. Dahl. Sequentially-allocated merge-split sampler for conjugate and nonconjugate Dirichlet process mixture models. *Submitted to Journal of Computational and Graphical Statistics*, 2005.

[7] D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixture models. *Bayesian Analysis*, 1(1):121–144, 2006.

[8] Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. In *NIPS*, 2008.

[9] K. Kurihara, M. Welling, and N. Vlassis. Accelerated variational Dirichlet process mixtures. In *NIPS*, 2006.

[10] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, 1999.

[11] O. Papaspiliopoulos and G. O. Roberts. Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models. *Biometrika*, 95(1):169–186, 2008.

[12] N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. Tenenbaum. Church: A language for generative models. In *Uncertainty in Artificial Intelligence*, 2008.

[13] N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, 2012.

[14] N. Ueda and Z. Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(1):1223–1241, 2002.

[15] C. Wang and D. Blei. Truncation-free stochastic variational inference for Bayesian nonparametric models. In *NIPS*, 2012.

[16] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[17] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[18] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[19] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.

[20] D. Zoran and Y. Weiss. Natural images, Gaussian mixtures and dead leaves. In *NIPS*, 2012.

Supplementary Material: Memoized Online Variational Inference for Dirichlet Process Mixture Models

Michael C. Hughes and Erik B. Sudderth
Department of Computer Science, Brown University, Providence, RI 02912
mhughes@cs.brown.edu, sudderth@cs.brown.edu

Abstract

This document contains supplementary mathematics and algorithm descriptions to help readers understand our new learning algorithm. First, in Sec. 1 we offer detailed model description and update equations for a DP-GMM with zero-mean, full-covariance Gaussian likelihood. Second, in Sec. 2 we provide step-by-step discussion of our birth move algorithm, providing a level-of-detail at which the interested reader could implement our approach.

1 DP mixtures with zero-mean Gaussian observations

To review, consider the generic DP mixture model defined in the main text.

$$G \sim \text{DP}(\alpha_0 H), \quad G \triangleq \sum_{k=1}^{\infty} w_k \delta_{\phi_k}, \quad v_k \sim \text{Beta}(1, \alpha_0), \quad w_k = v_k \prod_{\ell=1}^{k-1} (1 - v_\ell). \tag{1}$$

This process produces mixture weights w_k from a *stick-breaking* process and data-generating parameters ϕ_k from base measure H. Each data item n chooses an assignment $z_n \sim \text{Cat}(w)$, and then draws observations $x_n \sim F(\phi_{z_n})$. We assume both H and F belong to exponential families

$$p(\phi_k \mid \lambda_0) = \exp\Big\{\lambda_0^T t_0(\phi_k) - a_0(\lambda_0)\Big\}, \qquad p(x_n \mid \phi_k) = \exp\Big\{\phi_k^T t(x_n) - a(\phi_k)\Big\}. \tag{2}$$

We now make this process concrete, providing the complete model and variational approximation for the particular case where observed data consists of a length-D column vector x_n and the observation model $F(x|\phi_k)$ is zero-mean Gaussian.

1.1 Zero-mean Gaussian Observation Model $F(x|\phi_k)$

For the Gaussian case, we parameterize the Gaussian likelihood $F(x|\phi_k)$ for component k by a D-length mean vector μ_k and a $D \times D$ symmetric, positive definite precision matrix Λ_k. Let $\phi_k = (\mu_k, \Lambda_k)$. For the zero-mean likelihood, we assume $\mu_k = 0$ for all k. This leaves only precision matrix Λ_k as a parameter of interest. The likelihood of x_n when assigned to component k is

$$p(x_n|z_n = k) = \text{Normal}(x_n|0, \Lambda_k^{-1}) \tag{3}$$

$$\log p(x_n|z_n = k) = -\frac{D}{2}\log[2\pi] + \frac{1}{2}\log|\Lambda_k| - \frac{1}{2}x_n^T \Lambda_k x_n \tag{4}$$

$$= -\frac{D}{2}\log[2\pi] + \frac{1}{2}\log|\Lambda_k| - \frac{1}{2}\text{tr}(\Lambda_k x_n x_n^T) \tag{5}$$

where $|P|$ represents the *determinant* of a square matrix P.

Writing the quadratic form in terms of the trace function $\text{tr}(\cdot)$, which is a linear function, makes it clear that this distribution belongs to the exponential family, with sufficient statistic $t(x_n) = x_n x_n^T$. This follows from the identity $\text{tr}(AB) = \text{vec}(A)^T \text{vec}(B)$, where $\text{vec}(\cdot)$ vectorizes a $Q \times R$ matrix into a column vector of length $Q \cdot R$.

1.2 Wishart base measure $H(\phi_k)$

The conjugate base measure $H(\phi_k|\lambda_0)$ for this likelihood is the Wishart distribution. The parameters are $\lambda_0 = \nu, W$, where ν is a scalar degrees-of-freedom satisfying $\nu \geq D$, and W is a $D \times D$ symmetric, positive definite matrix.

$$p(\Lambda_k|\nu, W) = \text{Wish}(\nu, W) \tag{6}$$

$$\log p(\Lambda_k|\nu, W) = -\log \mathbb{Z}(\nu, W) + \frac{\nu - D - 1}{2} \log |\Lambda_k| - \frac{1}{2}\text{tr}(W^{-1}\Lambda_k) \tag{7}$$

$$\log \mathbb{Z}(\nu, W) = \frac{\nu D}{2} \log 2 + \log \Gamma_D\Big(\frac{\nu}{2}\Big) - \frac{\nu}{2} \log \Big|W^{-1}\Big| \tag{8}$$

where $\Gamma_D(a)$ is the multivariate Gamma function, defined as $\Gamma_D(a) = \pi^{D(D-1)/4} \prod_{d=1}^{D} \Gamma(a + \frac{1-d}{2})$

1.3 Variational Approximation

To approximate the full (but intractable) posterior over variables z, v, ϕ, we consider a fully-factorized variational distribution q, with individual factors from appropriate exponential families:

$$q(\mathbf{z}, v, \phi) = \prod_{n=1}^{N} q(z_n|\hat{r}_n) \prod_{k=1}^{K} q(v_k|\hat{\alpha}_1, \hat{\alpha}_0) q(\phi_k|\hat{\lambda}_k), \tag{9}$$

$$q(z_n) = \text{Cat}(z_n \mid \hat{r}_{n1}, \ldots \hat{r}_{nK}), \quad q(v_k) = \text{Beta}(v_k \mid \hat{\alpha}_{k1}, \hat{\alpha}_{k0}), \quad q(\phi_k) = H(\phi_k \mid \hat{\lambda}_k). \tag{10}$$

Local assignments $q(z_n)$ The posterior over assignments for each item n – $p(z_n|x_n, \phi, v)$ – is approximated by a discrete distribution over K components. Although the model allows assignment to an *unbounded* set, we enforce truncation $q(z_n > K) = 0$ to make inference tractable.

Parameters $\hat{r}_{n1} \ldots \hat{r}_{nK}$ for each $q(z_n)$ must be non-negative and sum-to-one. Each $\hat{r}_{nk}$ is interpreted as the fraction of posterior responsibility that component k has for x_n. Update equations are:

$$q(z_n) = \text{Cat}(\hat{r}_{n1}, \hat{r}_{n2}, \ldots \hat{r}_{nK}) \tag{11}$$

$$\tilde{r}_{nk} = \exp\Big(\mathbb{E}_q[\log w_k(v)] + \mathbb{E}_q[\log p(x_n \mid \phi_k)]\Big), \quad \hat{r}_{nk} = \frac{\tilde{r}_{nk}}{\sum_{\ell=1}^{K} \tilde{r}_{n\ell}}. \tag{12}$$

Given estimates $\hat{r}_n$ for the whole dataset, we compute sufficient statistics for component k:

$$\hat{N}_k \triangleq \mathbb{E}_q\Big[\sum_{n=1}^{N} z_{nk}\Big] = \sum_{n=1}^{N} \hat{r}_{nk}, \qquad s_k(\mathbf{x}) \triangleq \mathbb{E}_q\Big[\sum_{n=1}^{N} z_{nk} t(x_n)\Big] = \sum_{n=1}^{N} \hat{r}_{nk} x_n x_n^T, \tag{13}$$

Global stick-breaking parameters $q(v)$ Each stick-breaking fraction v_k is given an independent variational factor $q(v_k)$, with update equations

$$q(v_k) = \text{Beta}(\hat{\alpha}_{k1}, \hat{\alpha}_{k0}), \quad \hat{\alpha}_{k1} = 1 + \hat{N}_k, \qquad \hat{\alpha}_{k0} = \alpha_0 + \sum_{\ell=k+1}^{K} \hat{N}_\ell \tag{14}$$

Given $\hat{\alpha}_{k1}, \hat{\alpha}_{k0}$ for all components, we may compute expected log mixture weights

$$\mathbb{E}_q[\log v_k] = \psi(\hat{\alpha}_{k1}) - \psi(\hat{\alpha}_{k1} + \hat{\alpha}_{k0}) \qquad \mathbb{E}_q[\log 1 - v_k] = \psi(\hat{\alpha}_{k0}) - \psi(\hat{\alpha}_{k1} + \hat{\alpha}_{k0}) \tag{15}$$

$$\mathbb{E}_q\Big[\log w_k(v)\Big] = \mathbb{E}_q[\log v_k] + \sum_{\ell=1}^{k-1} \mathbb{E}_q[\log 1 - v_\ell] \tag{16}$$

where $\psi(a)$ is the digamma function, the first derivative of $\log \Gamma(a)$.

Global data-generation parameters We define a separate factor for each component's data-generating parameters $q(\Lambda_k)$, to approximate the posterior $p(\Lambda_k|\mathbf{x}, \mathbf{z}, \ldots)$. Each factor is Wishart with parameters $\hat{\nu}_k, \hat{W}_k$, updated as follows

$$q(\Lambda_k) = \text{Wishart}(\Lambda_k|\hat{\nu}_k, \hat{W}_k) \tag{17}$$

$$\hat{\nu}_k = \nu + \hat{N}_k, \qquad \hat{W}_k^{-1} = W^{-1} + s_k(\mathbf{x}) \tag{18}$$

Given $\hat{\nu}_k, \hat{W}_k^{-1}$, we compute the expected log probability under component k for each data item x_n

$$\mathbb{E}_q\Big[\log p(x_n|\phi_k)\Big] = -\frac{D}{2}\log[2\pi] + \frac{1}{2}\mathbb{E}_q\Big[\log|\Lambda_k|\Big] - \frac{1}{2}\text{tr}(\mathbb{E}_q[\Lambda_k]x_n x_n^T) \tag{19}$$

Here, we use basic expectations under the Wishart distribution:

$$\mathbb{E}_q[\Lambda_k] = \hat{\nu}_k\hat{W}_k, \qquad \mathbb{E}_q[\log|\Lambda_k|] = \psi_D\Big(\frac{\hat{\nu}_k}{2}\Big) + D\log 2 + \log|\hat{W}_k| \tag{20}$$

where $\psi_D(a) = \sum_{d=1}^{D}\psi(a + \frac{1-d}{2})$ is the multivariate digamma function of dimension D.

2 Birth Moves for Mixture Models

Overview. As input, our birth procedure takes an existing variational model q with K components, together with global sufficient statistics $S^0 = [S_1^0\ S_2^0\ \ldots\ S_K^0]$ for the full dataset $\mathbf{x}$. The algorithm consists of 3 steps: *collection* of a subsample dataset $\mathbf{x}'$, *creation* of brand-new components by a fresh DP mixture model variational analysis of $\mathbf{x}'$, and *adoption* of these fresh new components by the full dataset $\mathbf{x}$. The output will be an expanded model q^* with $K + J'$ components.

2.1 Collection of the target dataset $\mathbf{x}'$

We find it simplest to focus on a birth move which *targets* a specific component k'. After selecting the component k', the birth move proceeds to subsample data $\mathbf{x}'$ associated with k', using the existing local assignment factors $q(z_n)$ to identify which data items to subsample. Certainly other ways of subsampling exist, but this has an intuitive interpretation as targeting a single sub-optimal component which may be too coarse (explaining multiple ideal subclusters) and refining it.

Selecting the target component k'. The procedure for selecting which component k' to target is not complicated. For understanding the mechanics of birth moves, it is fine to simply select the component k' uniformly at random. If we have K active components in original model q, then

$$k' \sim \text{Unif}(\{1, 2, \ldots K\}) \tag{21}$$

Many other schemes for choosing k' can be considered. But the above is perfectly sufficient, albeit potentially slow at trying a diverse set of possible moves in a short timespan.

In practice, we recommend sampling k' at random, but in a way that *biases* towards choosing components that (1) have more mass and (2) have not been targeted in the last few moves. Let $\hat{N}_k^0$ give the current expected count on the full dataset, and L_k denote the number of passes through the data since component k was last chosen for a birth move.

$$p(k' = k) \propto (\hat{N}_k^0) * (L_k)^2 \tag{22}$$

Squaring the L_k term forces the algorithm to not wait very long between trying all possible components, ensuring good coverage of the space of all possible moves. We found that this revised selection procedure improved the speed with which our algorithm recovered all missing components, but uniform selection should eventually reach the same high-quality configurations.

Sampling a dataset targeted on component k'. After selecting k', next we collect a *targeted* dataset $\mathbf{x}'$ with size at most N'. We recommend choosing N' large enough that necessary "undiscovered" components (not in the existing set $\{1, 2, \ldots K\}$ can be learned, but still small enough that running many batch VB iterations does not take more than a few seconds. We found $N' = 10000$

to be a good choice for our experiments using Gaussian likelihoods with dimension $D = 25$ to $D = 50$. For small values like $D = 2$, N' in the low hundreds may be sufficient.

The target dataset $\mathbf{x}'$ contains samples without replacement from the full dataset $\mathbf{x}$ (of size N). For each observed vector $x_n \in \mathbf{x}$, we add it to our subsample $\mathbf{x}'$ if the following test is true:

$$\hat{r}_{nk'} > \tau, \quad \text{with typical value } \tau = 0.1 \tag{23}$$

Here, $\hat{r}_{nk'}$ is interpreted as the posterior responsibility of component k' for data item n. Each observation n has a vector $[\hat{r}_{n1}\ \hat{r}_{n2}\ \cdots\ \hat{r}_{nK}]$ of these responsibilities, where each entry is non-negative and the whole vector sums to one. The value $\hat{r}_{nk'} \in [0, 1]$ will be larger than the threshold τ if the n-th observation is well-explained by component k'.

Intuitively, our simple "threshold" test for adding data to the targeted dataset $\mathbf{x}'$ ensures that the subsample contains data which are significantly explained by component k', while also promoting diversity (since members could also be partially explained by some other component). The threshold of 0.1 strikes a good balance between these competing goals. We did explore a few other values for τ among $\{0.2, 0.5\}$ in preliminary experiments, and found that $\tau = 0.1$ performed slightly better. We stress that this does not need to be fine-tuned for the particular dataset at hand: the same setting was used for all our experiments.

In practice collection is done by visiting each batch in turn, and collecting all relevant data items until the size of $\mathbf{x}'$ exceeds the limit N'. When batch traversal order is randomized at each pass through the data, this has the beneficial effect of randomizing the subsample.

2.2 Creating an expanded model with brand-new components from the targeted dataset

Next, we consider adding new components to our existing model. We first train a fresh DP mixture model with K' brand-new components on $\mathbf{x}'$ via conventional (batch) variational inference, and then later combine these components with the existing K component model.

The process of creating components by a fresh variational analysis is general and elegant. This strategy applies to *any* DP mixture with exponential family likelihoods, re-uses existing code routines needed for the larger learning algorithm, and has a pleasing interpretation as a "divide-and-conquer" strategy. That is, to find the ideal clustering for the large dataset $\mathbf{x}$, we simply need to repeatedly find some broadly related subset $\mathbf{x}'$ and perform a more fine-grained clustering of that subset.

Creation of new components. Given the target dataset $\mathbf{x}'$ as a stand-alone dataset for analysis, we perform one run of standard full-dataset variational inference. We fit a K'-component DP mixture model with exactly the same prior parameters as the original model.

In practice, we initialize by setting fixed-truncation $K' = 10$, which is a reasonable compromise between diversity and speed. To initialize, we select K' observations (uniformly at random) from $\mathbf{x}'$ to seed parameters. We run only for a fixed budget of $I' = 100$ iterations or until convergence of the objective, whichever happens first.

The choices of truncation level K', initialization routine, and number of iterations I' may all impact the performance of the birth move. We found the same settings lead to reasonable performance across all tested datasets. In general, a more intelligent initialization is better. Running for longer will produce more refined components, but at the cost of increased run-time.

After the run, instead of saving estimated parameters we save *summaries* for each new component:

$$\hat{N}' = [\hat{N}_1\ \hat{N}_2\ \cdots\ \hat{N}_{K'}] \tag{24}$$

$$s(\mathbf{x}') = [s_1(\mathbf{x}')\ s_2(\mathbf{x}')\ \cdots\ s_{K'}(\mathbf{x}')] \tag{25}$$

In general, some final components may have very few assignments to data $\mathbf{x}'$. Some may be empty or nearly-empty. We thus post-process results to remove components j which have low expected counts $\hat{N}_j$ for explaining the data $\mathbf{x}'$. Pruning out empty components makes later phases much faster without sacrificing quality.

Specifically, we remove component j if $\hat{N}_j < \epsilon N'$, and we set $\epsilon = \frac{1}{20}$. After this removal, we end up with a set of J' sufficient statistics $\{\hat{N}_j, s_j(\mathbf{x}')\}_{j=1}^{J'}$, where $J' \leq K'$. These sufficient statistics are all we pass along to the next step.

If only $J' = 1$ component is left, by construction its summary will be very close to the summary for the target component k'. We shouldn't expect adding this new component will improve the original model (since k' already exists unchanged). Thus, if the resulting number of components is $J' = 1$, we abort the birth process early and return to the original K component model.

Creation of combined model Here, we combine the K components from the existing model with the brand-new J' components. Working purely in terms of sufficient statistics, we find that it is easy to build a coherent combined model simply by *concatenating* the fresh components $S' = [\hat{N}' \; s(\mathbf{x}')]$ onto the existing global sufficient statistics $S^0 = [\hat{N} \; s(\mathbf{x})]$.

We now have an expanded model with $K + J'$ summaries, $S^* = [\hat{N}^* s^*]$:

$$\hat{N}^* = [\hat{N}_1 \; \hat{N}_2 \; \cdots \; \hat{N}_K \;\; \hat{N}'_{K+1} \; \hat{N}'_{K+2} \; \cdots \; \hat{N}'_{K+J'}] \tag{26}$$

$$s^* = [s_1(\mathbf{x}) \; s_2(\mathbf{x}) \; \cdots \; s_K(\mathbf{x}) \;\; s_{K+1}(\mathbf{x}') \; s_{K+2}(\mathbf{x}') \; \cdots \; s_{K+J'}(\mathbf{x}')] \tag{27}$$

This concatenation creates a valid set of sufficient statistics for an "expanded" dataset formed by the union of $\mathbf{x}$ and $\mathbf{x}'$. This set "double-counts" the subsample $\mathbf{x}'$, assigning these data items to both original components (mostly k') and new components $K + 1, \ldots K + J$". In the next phase (adoption), we pass through the entire dataset, and discover which interpretation (original or new components) is preferred by the model.

Using new, expanded sufficient statistics S^*, we can then expand both local and global factors. The resulting expanded model q^* remains valid due to our *nested* truncation of the variational posterior. At this stage, no local parameters have been assigned to the new components. For all n, we simply expand $q^*(z_n)$ to be a discrete distribution over $K + J'$ components, where only the first K have mass:

$$\text{Before: } q(z_n) = \text{Cat}(\hat{r}_{n1}, \ldots \hat{r}_{nK}) \tag{28}$$

$$\text{After: } q^*(z_n) = \text{Cat}(\hat{r}_{n1}, \ldots \hat{r}_{nK}, 0, 0, \ldots 0) \tag{29}$$

Crucially, all parameters $\hat{r}_{nk}$ are directly transfered from the previous model q, and no batches actually need to be visited at this stage (they can instead be lazily expanded during each visit of the adoption pass). Another consequence of this construction is that $q^*(\phi_k) = q(\phi_k)$ for all original components $k = 1, 2, \ldots K$, including the target component k'. For new components j, $q^*(\phi_j)$ are set to the resulting factors from the targeted analysis.

Only the stick-breaking factors $q^*(v)$ must be completely re-written after the expansion. Expansion forces these factors to shift probability mass onto newly inserted components. Given the counts from all $K + J'$ summaries S^*, the update equations become

$$q^*(v_k)|S^* = \text{Beta}(v_k|\alpha^*_{k1}, \alpha^*_{k0}), \qquad \hat{\alpha}^*_{k1} = 1 + \hat{N}_k, \quad \hat{\alpha}^*_{k0} = \alpha_0 + \sum_{\ell=k+1}^{K+J'} \hat{N}_k \tag{30}$$

The choice to insert new components last in the stick-breaking order (which is implicitly done by concatenation) is fairly principled. On average, freshly discovered components will be more "rare" than the original ones, and so will likely have smaller effective mass. Since the stick-breaking construction is "size-biased", inserting components with smaller mass later in the order makes sense.

2.3 Adoption of the new components

After expanding the model to have $K + J'$ components, we then proceed normally through the memoized variational inference E-step (local factors) and M-step (global factors) at each batch. Our goal in this pass is to have newborn components become "adopted" by the original dataset $\mathbf{x}$, attaining critical mass by actively explaining some data in $\mathbf{x}$.

At the start of this pass, we have the expanded set of global sufficient statistics S^* described earlier. Retaining the target summaries S' as well as the previous global summaries S^0 within S^* allows each brand-new component a chance to influence several batches of data.

To understand this necessity, consider the alternative: after creating an expanded model, we discard S' and keep only the original summaries S^0. To be consistent with local assignments, we must

expand S^0 to have zero mass on new components $K+1, K+2, \ldots K+J'$. Now, imagine visiting the first batch of data $\mathcal{B}_1$ and performing the E-step. After this step, the only mass on new components will come from the current batch. For each new component j: $S_j^0 = S_j(\mathcal{B}_1)$. If this batch does not assign any mass to component j, then $s_j^0 = 0$. Next, the following M-step (see the main text's Eq. (9)) will completely rewrite $\hat{\lambda}_j = \lambda_0 + 0 = \lambda_0$, reseting to its prior value. Component j will lose all information from the targeted dataset $\mathbf{x}'$ after only one update, becoming useless even though later batches may have highly preferred it.

To avoid this disaster, we choose to retain the "dual" interpretations of the data $\mathbf{x}'$ in S^* throughout the pass, which ensures every brand-new component j always has mass at least $\hat{N}'_j$. Thus, even when the first batch is not assigned at all to component j, we'll have $s_j^* = s_j(\mathbf{x}')$, and the update $\hat{\lambda}_j = \lambda_0 + s_j(\mathbf{x}')$ will retain vital information from our targeted analysis.

At the end of the adoption pass, immediately before the last M-step update to global parameters, we subtract-away all targeted summaries S' from the final global summaries S^*. This ensures that by the end of the adoption pass, both the final global summary and all global factors $q^*(v), q^*(\phi)$ have scale exactly consistent with the dataset $\mathbf{x}$. Under these conditions, the ELBO can be calculated exactly and merges can proceed.

2.4 Multiple birth moves in one pass

As a final note, performing several birth moves during one pass (refining multiple components at once) is definitely possible. We need only to collect several subsampled datasets $\mathbf{x}'_1, \mathbf{x}'_2, \ldots$, discover new components from each one via separate variational analyses, and then adopt all new components into an expanded model. For simplicity we focus on just one birth in the description below. All our experiments perform just one birth per pass, except for the final analysis of 8×8 image patches, where we execute two births per pass.