Large-scale Distributed Dependent Nonparametric Trees

Zhiting Hu ZHITINGH@CS.CMU.EDU
Language Technologies Institute, Carnegie Mellon University

Qirong Ho HOQIRONG@GMAIL.COM
Institute for Infocomm Research, A*STAR

Avinava Dubey AKDUBEY@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University

Abstract

Practical applications of Bayesian nonparametric (BNP) models have been limited, due to their high computational complexity and poor scaling on large data. In this paper, we consider *dependent nonparametric trees* (DNTs), a powerful infinite model that captures time-evolving hierarchies, and develop a large-scale distributed training system. Our major contributions include: (1) an effective memoized variational inference for DNTs, with a novel birth-merge strategy for exploring the unbounded tree space; (2) a model-parallel scheme for concurrent tree growing/pruning and efficient model alignment, through conflict-free model partitioning and lightweight synchronization; (3) a data-parallel scheme for variational parameter updates that allows distributed processing of massive data. Using 64 cores in 36 hours, our system learns a 10K-node DNT topic model on 8M documents that captures both high-frequency and long-tail topics. Our data and model scales are orders-of-magnitude larger than recent results on the hierarchical Dirichlet process, and the near-linear scalability indicates great potential for even bigger problem sizes.

1. Introduction

Bayesian nonparametric (BNP) methods provide a powerful framework for learning the internal structure of data. For example, Dirichlet processes can be used to cluster with an unbounded number of centers; its derivatives are used in applications such as storyline tracking (Ahmed et al., 2011), taxonomy induction (Ho et al., 2012), and image segmentation (Sudderth & Jordan, 2009), to name a few. Recently, a flexible dependent nonparametric trees (DNTs) model (Dubey et al., 2014) was developed to induce hierarchical structures with unbounded width and depth, which in turn supports cluster creation, evolution, and extinction over time.

Despite their flexibility and expressiveness, practical applications of BNP models have unfortunately been limited due to their high computational complexity and poor scaling on large data. While there has been recent work on efficient parallel inference algorithms for BNPs, the problem scales were limited to 100s to 1000s of samples, and at most 100s of components with a few million parameters (Table 1). This stands in stark contrast to recent industrial-scale algorithms for parametric models (e.g. the LDA topic model (Blei et al., 2003)), which typically handle billions of samples, millions of components, and trillions of parameters (Yuan et al., 2015). Such big parametric models have been shown to capture *long-tail* semantics (Wang et al., 2014) that improve the performance of industrial applications, yet this long-tail capability is, arguably, better realized using BNP models, with their unlimited and self-adjusting model capacity. Hence, we believe there is an urgent need to develop highly-scalable solutions for BNPs, in order to boost their utility on real-world problems.

A second issue is that most existing parallel BNP research is focused on the (hierarchical) Dirichlet process, and it is unclear how those techniques might be applied to other BNP models, including the tree-structured models considered in this paper (which are desirable for organizing information into easily-navigated hierarchies (Ghahramani et al., 2010; Blei et al., 2010)). The main issue is that tree models contain additional complex dependencies which render the inference hard to parallelize, while the changing tree structures can impose costly alignment overheads between parallel workers (Pan et al., 2013).

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

To address these challenges, we present a distributed large-scale training system for the DNTs model. On the algorithm side, we develop an efficient *memoized* variational inference (VI) algorithm (Hughes & Sudderth, 2013; Neal & Hinton, 1998) for DNTs, which incrementally tracks compact summary statistics of the full dataset through simple accumulation operations, and uses them to update tree node model parameters in a manner that is learning-rate-free and insensitive to batch size (which are advantages over stochastic VI (Hoffman et al., 2013)). To ensure unbounded tree width and depth (needed to fully explain large data), we devise a novel birth-merge strategy (Bryant & Sudderth, 2012; Jain & Neal, 2004) to allow adaptive creation and pruning of tree nodes.

This tree-structured inference algorithm is executed over a distributed cluster through a combination of data- and model-parallelism, interleaved in a manner that maximizes concurrency among workers. Specifically, our system alternates between two algorithm phases: the first phase is data-parallel *parameter learning*, where the tree layout is held fixed, and workers independently process data batches in order to update each tree node's parameters via a parameter server[1]. In the second phase, model-parallel *structure updating* (in which we update the tree layout), we logically partition the whole tree across workers, and workers concurrently apply birth-merge operations on their model parts. Here, we design a communication-efficient strategy for synchronizing tree structures across workers, where only lightweight operation records are exchanged over the network.

Our VI algorithm and distributed parameter server implementation is able to train large-scale DNTs models: Using 64 cores in 36 hours, our system can process over 8M documents to learn a truncation-free topic tree with 10K topics, in order to capture long-tail semantics. As shown in Table 1, in terms of data and parameter size, our result is orders-of-magnitude larger even when compared to recent results on the inferentially simpler hierarchical Dirichlet process (HDP). To the best of our knowledge, the scales achieved by our system constitute a new record for tree-structured BNP learning. We believe our distributed training strategies, including the conflict-free model alignment scheme and lightweight record communication, as well as the combined data- and model-parallelism, could be useful for scaling up other BNPs.

System	Smyth et al. 2009	Williamson et al. 2013	Chang & Fisher III 2014	Bryant & Sudderth 2012	Ours
Model	HDP	HDP	HDP	HDP	DNTs
Infer alg	MCMC	MCMC	MCMC	VI	VI
Single-machine parallel	√	√	√	·	√
Multi-machine parallel	√	·	·	·	√
Data size #doc #token	1.5K 2M	2.5K 3M	0.3M 100M	1.8M 120M	**8.4M** **720M**
Model size #topic #param	800 5.6M	80 1.2M	200 20M	600 4.8M	**10K** **700M**
Train time	–	7.5hr	28hr	40hr	36hr

Table 1. Comparison of recent BNP training frameworks. Results are quoted from the respective papers.

2. Dependent Nonparametric Trees

Real-world data usually contains rich hierarchical structures (Ghahramani et al., 2010) and exhibits vibrant variation (Ahmed et al., 2011). E.g., some news articles may be related to a general topic "sports" while some others are about a more specific topic "football". Additionally, along the time topics evolve and new topics merge.

To model the time-evolving hierarchically structured data, Dubey et al. 2014 proposed the dependent nonparametric trees (DNTs), a dependent nonparametric process with tree marginals known as the tree-structured stick breaking process (TSSBP) (Ghahramani et al., 2010). TSSBP is an infinite tree, where each node ϵ is associated with a mass π_ϵ such that $\sum_\epsilon \pi_\epsilon = 1$. Each data point is assigned to any of the infinitely many internal nodes in the tree, according to $p(z_n = \epsilon) = \pi_\epsilon$. The TSSBP can be represented using two interleaving stick-breaking processes – one (parameterized by α) that determines the size of a node and the other (parameterized by γ) that determines the branching probabilities. Index the root node as node $\emptyset$, its (infinite) child nodes as node 1, node $2, \ldots$, and the child nodes of node 1 as node $1 \cdot 1$, node $1 \cdot 2, \ldots$, etc. Then the infinite-dimensional tree is sampled as follows:

$$\nu_\epsilon \sim \text{Beta}(1, \alpha), \;\; \psi_\epsilon \sim \text{Beta}(1, \gamma), \;\; \pi_\emptyset = \nu_\emptyset, \;\; \phi_\emptyset = 1,$$
$$\phi_{\epsilon \cdot i} = \psi_{\epsilon \cdot i} \textstyle\prod_{j=1}^{i-1}(1 - \psi_{\epsilon \cdot j}), \quad \pi_\epsilon = \nu_\epsilon \phi_\epsilon \textstyle\prod_{\epsilon' \prec \epsilon}(1 - \nu_{\epsilon'})\phi_{\epsilon'},$$

where $\epsilon' \prec \epsilon$ indicates that ϵ' is an ancestor of ϵ.

The temporal variation of the clustering includes two aspects: the sizes of the clusters vary over time, and the locations of clusters in parameter space vary over time. For the former, the latent variables $\nu_\epsilon, \psi_\epsilon$ and π_ϵ are replaced with sequences $\nu_\epsilon^{(t)}, \psi_\epsilon^{(t)}$ and $\pi_\epsilon^{(t)}$ indexed by discrete time $t \in \mathcal{T}$. Let $N^{(t)}$ be the number of observations at time t; $z_n^{(t)}$ be the node assignment of the nth observation at t; $X_\epsilon^{(t)} = \sum_{n=1}^{N^{(t)}} \mathbb{I}(z_n^{(t)} = \epsilon)$ be the number of observations assigned to node ϵ at t; and $Y_\epsilon^{(t)} = \sum_{n=1}^{N^{(t)}} \mathbb{I}(\epsilon \prec z_n^{(t)})$ be the number of observations assigned to the descendants

[1] We use the Petuum parameter server (Ho et al., 2013; Dai et al., 2015) from `petuum.org`.

of ϵ. Then the prior predictive distribution over the tree at time t is defined as:

$$\nu_\epsilon^{(t)} \sim \text{Beta}(1+\sum\nolimits_{t'} X_\epsilon^{(t')}, \alpha+\sum\nolimits_{t'} Y_\epsilon^{(t')}),$$
$$\psi_{\epsilon\cdot i}^{(t)} \sim \text{Beta}(1+\sum\nolimits_{t'} X_{\epsilon\cdot i}^{(t')}+Y_{\epsilon\cdot i}^{(t')}, \gamma+\sum\nolimits_{j>i,t'} X_{\epsilon\cdot j}^{(t')}+Y_{\epsilon\cdot j}^{(t')}),$$

where $t' \in \mathbb{N}$ ranges from $t-h$ to $t-1$, with $h \in \mathbb{N}$ being a "window" parameter. Refer to the sequence of trees as $(\Pi^{(t)} = ((\pi_\epsilon^{(t)}), (\phi_{\epsilon\cdot i}^{(t)})), t \in \mathcal{T})$. It is provable that the marginal posterior distribution of the above dynamic TSSBP at time t follows a TSSBP, which implies the above equation defines a *dependent TSSBP*.

Each node is associated with a parameter value $\theta_\epsilon^{(t)}$ to model the observed data. Intuitively, within a tree $\Pi^{(t)}$, nodes have similar values to their parents; and between trees $\Pi^{(t-1)}$ and $\Pi^{(t)}$, corresponding parameters $\theta_\epsilon^{(t-1)}$ and $\theta_\epsilon^{(t)}$ have similar values. If the data is document, the DNTs can be used to model evolving hierarchical topics.

Time-varying hierarchical topic models Given a vocabulary of V words, a document can be represented as a V-dimensional term frequency vector, that corresponds to a location on the surface of the $(V-1)$-dimensional unit sphere. The von Mises-Fisher (vMF) distribution of order V provides a probability density on this space:

$$f(\boldsymbol{x};\boldsymbol{\mu},\kappa) = C_V(\kappa)\exp(\kappa\boldsymbol{\mu}^\top\boldsymbol{x}); C_V(\kappa) = \frac{\kappa^{.5V-1}}{(2\pi)^{.5V} I_{.5V-1}(\kappa)}.$$

where parameter $\boldsymbol{\mu}$ defines the mean direction; κ is the concentration parameter; and $I_v(a)$ is the modified bessel function of first kind. A mixture of vMF distributions can therefore be used to cluster documents, and the mean direction $\boldsymbol{\mu}_k$ of the kth cluster can be interpreted as the kth topic.

The mean parameter $\theta_{\epsilon\cdot i}^{(t)}$ of each node $\epsilon\cdot i$ is assigned as:

$$\theta_{\epsilon\cdot i}^{(t)} | \theta_\epsilon^{(t)}, \theta_{\epsilon\cdot i}^{(t-1)} \sim \text{vMF}(\tau_{\epsilon\cdot i}^{(t)}, \rho_{\epsilon\cdot i}^{(t)}), \tag{1}$$

where, $\rho_{\epsilon\cdot i}^{(t)} = \kappa_0 \| \kappa_1 \theta_\epsilon^{(t)} + \kappa_2 \theta_{\epsilon\cdot i}^{(t-1)} \|$, $\tau_{\epsilon\cdot i}^{(t)} = \frac{\kappa_0\kappa_1\theta_\epsilon^{(t)} + \kappa_0\kappa_2\theta_{\epsilon\cdot i}^{(t-1)}}{\rho_{\epsilon\cdot i}^{(t)}}$. The root node $\emptyset^{(t)}$ is assumed to have a pseudo-parent node with mean parameter $\theta_{-1}^{(t)}$ set to the centroid of the data. Each document $\boldsymbol{x}_n^{(t)}$ is sampled according to $z_n^{(t)} \sim \text{Discrete}(\Pi^{(t)})$ and $\boldsymbol{x}_n^{(t)} \sim \text{vMF}(\theta_{z_n}^{(t)}, \beta)$.

Parallel MCMC for online inference At each time t the parameters of tree $\Pi^{(t)}$ is inferred from data samples of this epoch, conditioned on the previous trees which are not relearned by backwards passes. Dubey et al. 2014 developed a parallel Gibbs sampler for tree parameters ($\Pi^{(t)}$) and topic indicators ($z_n^{(t)}$), and a MAP estimator for the location parameters ($\theta_\epsilon^{(t)}$). The parallelization is due to the fact that the tree parameters $\Pi^{(t)}$ are conditionally independent given $\{z_n^{(t)}\}$ (and historical $\{z_n^{(t')}\}$), and vice versa.

However, despite the parallelizability of the sampling procedures, it can be very costly to align the tree structures across processors. To avoid this cost, in practice, the parallel MCMC algorithm maintains a single copy of the tree model, on which all processors perform the sampling concurrently. This scheme in turn leads to heavy lock contentions between processors, as the shared model keeps creating and deleting nodes throughout sampling. This can severely deteriorates the scalability of the inference, as shown in our experiments.

3. Truncation-free Memoized VI for DNTs

Distributed learning for BNPs can be difficult: on the algorithmic side, correct parallelization of learning algorithms requires appropriate independence structure among variables. On the implementation side, efficient model state synchronization is necessary due to slow network communication. These issues are made even more challenging in the context of tree-structured DNTs, which present more complex inter-variable dependencies; consequently, model alignment between machines is very communication- and computation-expensive.

Variational techniques are an appealing starting point for parallelization, because they introduce model independencies that form natural boundaries for model-parallelism. In this section, we shall develop a highly-parallelizable variational inference (VI) algorithm for DNTs, and introduce an accompanying data- and model-parallelization strategy in the next section. Our key technical contributions are: (a) we develop a *memoized* VI technique for DNTs that are suitable for the distributed setting; (b) we design a novel birth-merge strategy to adaptively update tree structures and enable unbounded width and depth.

3.1. Parameter Learning

Memoized VI has been shown to be effective on Dirichlet process mixtures. It maintains a compact summary statistics of full dataset, and updates model parameters by replacing old statistics with new one when visiting a data batch. The additivity of the summaries allows the variational updates from multiple machines to be efficiently accumulated. Hence the best of the two previous VI algorithms are combined: the efficiency of the stochastic VI by interleaving global and local updates frequently, and the robustness of full-data VI by avoiding learning rates.

We first describe the variational updates for the vMF-based DNTs (details are given in the supplements), then present how the memoized inference proceeds. Here we focus on a specific time t and infer the parameters of $\Pi^{(t)}$ from the data samples of this epoch. We omit the label t in the following when there is no confusion.

Consider a family of factorized variational distributions:

$$q(\boldsymbol{\nu},\boldsymbol{\psi},\boldsymbol{\theta},\boldsymbol{z}) = \prod\nolimits_\epsilon q(\nu_\epsilon|\delta_\epsilon) q(\psi_\epsilon|\sigma_\epsilon) q(\theta_\epsilon|\mu_\epsilon,\eta_\epsilon) \prod\nolimits_n q(z_n|\lambda_n),$$

and assume the factors have the parametric forms:

$$q(\nu_\epsilon|\delta_\epsilon) = \text{Beta}(\nu_\epsilon|\delta_\epsilon), \quad q(\psi_\epsilon|\sigma_\epsilon) = \text{Beta}(\psi_\epsilon|\sigma_\epsilon),$$
$$q(\theta_\epsilon|\mu_\epsilon,\eta_\epsilon) = \text{vMF}(\theta_\epsilon|\mu_\epsilon,\eta_\epsilon), \quad q(z_n|\lambda_n) = \text{Multi}(z_n|\lambda_n).$$

To tractably handle the infinite set of components, we employ a *nested* truncation: suppose we have a truncated tree T (with finite number of nodes), then $q(z_n = \epsilon) = 0$ for $\epsilon \notin T$. This forces all data to be explained by only the nodes in T, while other nodes can be ignored. The truncation is nested such that if a truncated tree T' is a subtree of T then variational distributions over T' are a special case of those over T. Growing T therefore always gives potentially better solutions.

Variational algorithms aim to minimize KL divergence from q to the true posterior, which is equivalent to tightening the evidence lower bound (ELBO):

$$\mathcal{L}(q) = \sum\nolimits_{\epsilon\in T} \mathbb{E}\left[\log \frac{p(\nu_\epsilon)p(\psi_\epsilon)p(\theta_\epsilon)}{q(\nu_\epsilon)q(\psi_\epsilon)q(\theta_\epsilon)}\right] + \sum\nolimits_{n=1}^{N} \mathbb{E}\left[\log \frac{p(x_n|\theta_{z_n})p(z_n|\boldsymbol{\nu},\boldsymbol{\psi})}{q(z_n)}\right].$$

We optimize $\mathcal{L}(q)$ via coordinate ascent, iteratively updating the local parameters (data assignments $\{\lambda_n\}$) and global parameters (stick-breaking parameters $\{\delta_\epsilon, \sigma_\epsilon\}$ and data generating parameters $\{(\mu_\epsilon, \eta_\epsilon)\}$).

For observation x_n, the optimal solution of λ_n is given by:

$$\lambda^*_{n\epsilon} \propto \exp\left\{\mathbb{E}[\log\ p(z_n = \epsilon|\boldsymbol{\nu},\boldsymbol{\psi})] + \mathbb{E}[\log\ p(x_n|\theta_\epsilon)]\right\}. \quad (2)$$

To optimize global parameters of each component ϵ, we store its expected mass M_ϵ and sufficient statistics s_ϵ:

$$M_\epsilon \triangleq \mathbb{E}\left[\sum\nolimits_n z_{n\epsilon}\right] = \sum\nolimits_n \lambda_{n\epsilon},$$
$$s_\epsilon \triangleq \mathbb{E}\left[\sum\nolimits_n z_{n\epsilon}x_n\right] = \sum\nolimits_n \lambda_{n\epsilon}x_n. \quad (3)$$

The global parameters (at time t) can be updated based on only these (and historical) summaries. In particular, for the stick-breaking parameters:

$$\delta^{(t)*}_{\epsilon,1} = 1 + \sum_{t'=t-h}^{t} M^{(t')}_\epsilon, \quad \delta^{(t)*}_{\epsilon,2} = \alpha + \sum_{t'=t-h}^{t}\sum_{\epsilon\prec\epsilon'} M^{(t')}_{\epsilon'},$$
$$\sigma^{(t)*}_{\epsilon\cdot i,1} = 1 + \sum_{t'=t-h}^{t}\sum_{\epsilon\preceq\epsilon'} M^{(t')}_{\epsilon'}, \quad \sigma^{(t)*}_{\epsilon\cdot i,2} = \gamma + \sum_{t'=t-h}^{t}\sum_{j>i} M^{(t')}_{\epsilon\cdot j}. \quad (4)$$

Next, optimizing $\mathcal{L}(q)$ w.r.t $q(\theta^{(t)}_\epsilon)$, we have:

$$\log\ q^*(\theta^{(t)}_\epsilon) = \mathbb{E}[\rho^{(t)}_\epsilon \tau^{(t)\top}_\epsilon \theta^{(t)}_\epsilon] + \sum\nolimits_{\epsilon\prec\epsilon\cdot i} \mathbb{E}[\log\ C_V(\rho^{(t)}_{\epsilon\cdot i}) + \rho^{(t)}_{\epsilon\cdot i}\tau^{(t)\top}_{\epsilon\cdot i}\theta^{(t)}_{\epsilon\cdot i}] + \sum\nolimits_n \lambda_{n\epsilon}\mathbb{E}[\beta\theta^{(t)\top}_\epsilon x_n] + \text{const},$$

where $\rho^{(t)}_{\epsilon\cdot i}$ and $\tau^{(t)}_{\epsilon\cdot i}$ are defined in Eq.1 which encode the time- and hierarchical-dependency of the nodes in the dependent trees. With the complex dependency, the normalizer term $\mathbb{E}[\log\ C_V(\rho^{(t)}_{\epsilon\cdot i})]$ becomes intractable, we therefore turn to a *Taylor approximation*, which leads to the updates:

$$\mu^{(t)*}_\epsilon \approx \frac{\kappa_0(\kappa_1\mu^{(t)}_{\epsilon'} + \kappa_2\mu^{(t-1)}_\epsilon + \sum_i \kappa_1\mu^{(t)}_{\epsilon i} + a\mu^{(t-1)}_{\epsilon i}) + \beta s_\epsilon}{\eta^{(t)*}_\epsilon}$$
$$\eta^{(t)*}_\epsilon \approx \|\kappa_0(\kappa_1\mu^{(t)}_{\epsilon'} + \kappa_2\mu^{(t-1)}_\epsilon + \sum_i \kappa_1\mu^{(t)}_{\epsilon i} + a\mu^{(t-1)}_{\epsilon i}) + \beta s_\epsilon\|, \quad (5)$$

where $a = \frac{3(.5V-1)\kappa_0\kappa_1\kappa_2}{\bar{\rho}^{(t)}_{\epsilon i}}$, and $\bar{\rho}^{(t)}_{\epsilon\cdot i}$ is the same with $\rho^{(t)}_{\epsilon\cdot i}$ except that $\theta^{(t)}_\epsilon$ is replaced with the mean of the variational distribution over $\theta^{(t)}_\epsilon$ from the previous iteration.

Memoized inference Assume the data is divided into B fixed batches $\{\mathcal{B}_b\}$, the memoized VI tracks the statistics for each batch $S^b_\epsilon = (M^b_\epsilon, s^b_\epsilon)$, as well as the full-dataset statistics $S_\epsilon = (M_\epsilon, s_\epsilon)$. Note that the *additivity* of the sufficient statistics implies that the full-data statistics can be obtained exactly as the addition of summaries of distinct batches. Hence the memoized inference proceeds by visiting each batch $\mathcal{B}_b$, updating local parameters for that batch via Eq.2, and then updating the full-data summaries for each component by replacing the old statistics of $\mathcal{B}_b$ with new one:

$$S_\epsilon \mathrel{-}= S^b_\epsilon, \quad S^b_\epsilon \leftarrow \left(\sum_{n\in\mathcal{B}_b}\lambda_{n\epsilon}, \sum_{n\in\mathcal{B}_b}\lambda_{n\epsilon}x_n\right), \quad S_\epsilon \mathrel{+}= S^b_\epsilon. \quad (6)$$

Given the new full-data statistics, the global parameters can then be updated via Eqs.4-5.

Though the summary statistics require additional memory, they are compact (at the level of batch, rather than sample) and highly memory-efficient. Moreover, our algorithm only requires sequential access, as apposed to random access, to data batches, indicating the data batches (and batch statistics) can be stored in disks. This is also necessary for handling large data that cannot fit in the memory.

3.2. Tree Structure Updating

We next develop a data-driven birth-merge strategy to dynamically expand and contract the model size. The birth and merge moves are interleaved with the above variational updates, and are highly parallelizable as each move only involves a local region of the tree. The nested variational distributions ensure the stability of the inference.

3.2.1. Birth Move to Create New Nodes

Analog to the stick-breaking process, a birth move creates a new child for an existing node and appends it to the child

list. The move proceeds as follows: (1) select a target node ϵ; (2) collect a set of data samples $\mathcal{X}_\epsilon$ mainly explained by ϵ; (3) assuming node ϵ has $k-1$ children, create a new child $\epsilon \cdot k$, and estimate $\epsilon \cdot k$'s variational posteriors by a *restricted iteration* on $\mathcal{X}_\epsilon$; (4) adopt the new node.

Target selection Intuitively, a cluster associated with larger mass is more likely to contain sub-clusters. We therefore sample a target node ϵ according to the mass M_ϵ.

Data collection We collect a subset of data samples where each sample n has $\lambda_{n\epsilon} > 1/\sqrt{K}$. Here K is the number of existing components. The collected subset is highly related to the target node, and helps to better estimate the new child's parameter. Note that the collecting is done when passing through the data in parameter learning phase, and introduces little computational overhead.

Child creation In a hierarchical clustering, a high-quality child cluster is expected to be closely related to its parent, while differing from its siblings. We therefore initialize the variational mean of the vMF emission as $\mu_{\epsilon\cdot k} = \overline{\mu_\epsilon - \overline{\sum_j \mu_{\epsilon\cdot j}}}$, where $\overline{a}$ represents the L2-normalization of a. (If ϵ has no existing child, then $\mu_{\epsilon\cdot k}$ is initialized by the sample in $\mathcal{X}_\epsilon$ which allocates smallest mass on ϵ.) We then update the variational posteriors of node ϵ and $\epsilon\cdot k$ through a restricted VI on data $\mathcal{X}_\epsilon$ (Eqs.2-5) until convergence, while parameters of other nodes are held constant.

Child adoption We always accept the new child without assessing the change of ELBO produced by the insertion, and rely on subsequent merge moves to remove redundant nodes. Since the summary statistics S_ϵ and $S_{\epsilon\cdot k}$ were updated based on the target data collection $\mathcal{X}_\epsilon$, data samples in $\mathcal{X}_\epsilon$ are now counted twice in the global summary. We subtract away $\mathcal{X}_\epsilon$'s statistics after the new-born node $\epsilon \cdot k$ has been better adapted to the data through the normal variational updates.

A birth operation only requires write access to a node (and read access to its children). Thus it is straightforward to do birth moves concurrently on different parts of the tree.

3.2.2. Merge Move to Remove Redundant Nodes

Merge moves delete unneeded nodes and keep the model compact and efficient. A merge procedure consists of two steps: (1) select a pair of candidate nodes; (2) if the merge improves the ELBO, accept the merge.

Candidate selection In the tree-structured clustering, redundancy usually exists between two sibling clusters, or a child with its parent. We first randomly choose a node ϵ_a, then, from its siblings and parent, randomly select the other node ϵ_b by the cosine similarity between their variational means of vMF emissions: $\text{sim}(\epsilon_a, \epsilon_b) := \mu_{\epsilon_a}^\top \mu_{\epsilon_b}$. Note that the selection does not rely on node mass, since small-mass component may account for long-tail data.

Merge adoption The additivity (Eq.3) allows the merged global statistics to be constructed directly. The difference in the *full-data* ELBO by the merge can be approximately computed in constant time using only the node pair's global statistics (see the supplements for details). If the merge is accepted, the child nodes of ϵ_a are transferred to node ϵ_b.

A sibling-sibling merge only modifies two nodes of the same depth, while a parent-child merge will involve two nodes of different depths. In the next, we design specialized tree partition methods to parallelize the merge moves.

4. Distributed System Implementation

We now present a distributed framework that supports large-scale training for the DNTs model. Earlier, we described a VI algorithm for DNTs, which consists of two phases. In the parameter learning phase, the additivity of the summary statistics (Eq.3) suggests that, parameter updates from distinct workers for the same nodes can be conveniently accumulated, which accelerates convergence if workers process different data batches concurrently. Moreover, since the tree structure is fixed, the expensive tree alignment is avoided even if different workers hold their own model copies. This naturally leads to *data parallelism* for this phase.

On the other hand, the structure updating phase changes the tree structure by adding/removing nodes. By noting that both birth and merge moves only access to local regions of the whole tree, we can safely partition the tree into multiple parts and perform birth-merge moves on each part independently. This leads to *model parallelism* for this phase.

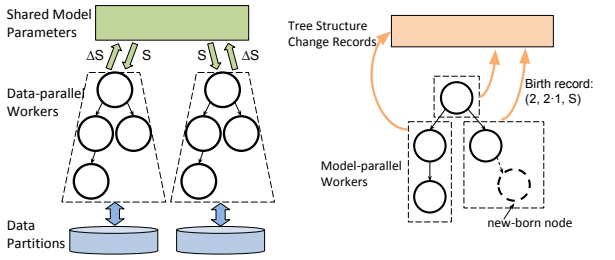
The two parallel schemes are interleaved, where workers are independent with each other in each phase. This ensures high degree of parallelism and scalability.

We build our system on top of an open-source distributed parameter server Petuum (`petuum.org`). Essentially, a parameter server (PS) presents a distributed shared memory interface, where worker machines can read/write any parameter, agnostic to the physical location of the parameter. Petuum provides bounded-asynchronous consistency, which reduces inter-iteration parameter synchronization times through a staleness parameter s.

Algorithm 1 summarizes the distributed training algorithm for DNTs. Next we describe the parallel schemes in detail.

4.1. Data Parallelism for Parameter Learning

Figure 1(a) shows the data parallel scheme. We use the PS to store the full-data statistics $\{S_\epsilon = (M_\epsilon, s_\epsilon)\}$, because of its additivity and easy restoration of other model parameters based on them (Eqs.4-5). The whole dataset is partitioned and distributed across all workers. Each worker only ever processes its assigned data shard.



(a) Data parallelism. (b) Model parallelism.

Figure 1. Architecture: data parallelism in parameter learning phase and model parallelism in structure updating phase.

Each worker holds a local copy of the full-data statistics, based on which the variational updates of Eqs.2-5 are conducted. After processing a data batch $\mathcal{B}_b$, the worker sends the difference of the batch statistics $\{(S_\epsilon^{b,new} - S_\epsilon^{b,old})\}$ to the PS, where the global statistics are updated by simple accumulation. This completes an update of the memoized VI (Eq.6). The worker then fetches the latest global statistics from the PS to replace its local copy, whereby the updates from other workers are synchronized. The communication overhead of global updates is linear to the model size.

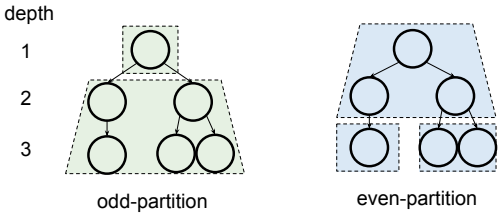


Figure 2. Odd- and even-partition for distributed merge moves. In both cases, siblings are guaranteed to be in the same parts.

4.2. Model Parallelism for Structure Updating

Figure 1(b) shows the model parallel scheme. The whole tree is partitioned into multiple parts, and each worker takes charge of several pieces to perform birth/merge operations. In practice, there are two key features to be carefully designed: (1) how to partition the tree to minimize dependency between different workers; (2) how to broadcast the local structure modification across all workers for alignment.

Conflict-free model partition Note that each worker *physically* holds a whole model copy (for data parallelism), so we only need to partition the tree *logically*, without actual transmission of model parts. Since different structure operations have different locality, we design distinct partition strategies for each of them. For birth moves, the partition simply guarantees each node is assigned to only one worker (e.g., Figure 1(b)). To avoid duplicated merge (i.e., one node being merged with multiple others by different workers), we divide merge into two modes: the *odd*-mode only allows merging between odd-depth nodes and their parents, and between siblings. The tree is accordingly partitioned such that every odd-depth node is assigned to the same workers with their parents, and all siblings are assigned to the same workers (Figure 2). In contrast, *even*-mode allows merging between even-depth child nodes and their parents, and between siblings. The tree is partitioned accordingly. birth and odd-/even-mode merge are three types of structure updating phases, and are interleaved with the parameter learning phase throughout training (Algorithm 1).

Lightweight structure synchronization To synchronize the tree structures, one straightforward method is that each worker broadcasts its assigned tree parts, to replace the corresponding parts of all local model copies in other workers. This, however, can cause costly communication overhead. We adopt an alternative strategy that workers exchange the *operation records*, based on which all workers can update their local model and produce new trees with exactly the same structure. For birth move, a record is $(\epsilon, \epsilon{\cdot}k, S_\epsilon, S_{\epsilon\cdot k})$, where ϵ is the parent node index, $\epsilon \cdot k$ is the new-born child node, and S_ϵ and $S_{\epsilon\cdot k}$ are the summary statistics after restricted updates; for merge move, a record is (ϵ_a, ϵ_b), indicating node ϵ_a is merged into ϵ_b. The records are therefore lightweight and can be shared across workers via PS efficiently. Also note that though every worker has to perform all structure updates, the operations are efficient as no restricted iterations (for birth), or objective evaluation (for merge) are needed.

Algorithm 1 Distributed training for DNTs

% line 7,17: Data-parallel VI
% line 8-13: Model-parallel (odd/even) merge
% line 16,18-21: Model-parallel birth
Train:
Initialize $\{\Pi, \mu\}$ randomly
repeat
  Memoized VI
  Sample (odd/even) merge pairs $\mathcal{P} \in$ assigned model part
  for all $(\epsilon_a, \epsilon_b) \in \mathcal{P}$ **do**
    if $\mathcal{L}^{merge} > \mathcal{L}$ **then**
      Send merge record to PS
    end if
  end for
  Read all merge records from PS
  Update local model structure
  Sample birth nodes $\mathcal{Q} \in$ assigned model part
  Memoized VI and collect target data
  for all $\epsilon \in \mathcal{Q}$ **do**
    Restricted update ϵ and $\epsilon \cdot k$
    Send birth record to PS
  end for
  Read all birth records from PS
  Update local model structure
until convergence

Data-parallel memoized VI:
for all batch $\mathcal{B} \in$ assigned data partition **do**
  Update λ_n for $n \in \mathcal{B}$, using Eq.2
  Compute new batch statistics $\{S_\epsilon^{b,new}\}$
  Send updates $\{S_\epsilon^{b,new} - S_\epsilon^{b,old}\}$ to PS
  Read global statistics from PS
  Update global parameters, using Eqs.4-5
end for

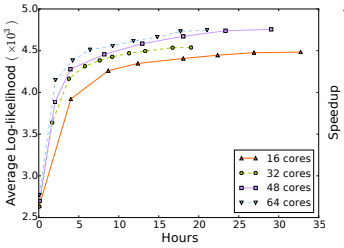


Figure 3. Converg on PubMed

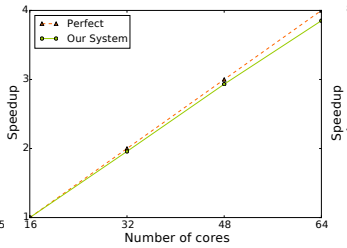


Figure 4. Speedup on PubMed

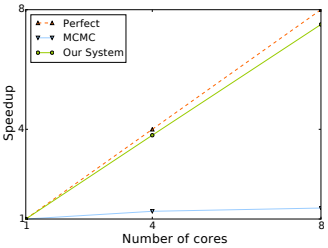


Figure 5. Speedup on PNAS

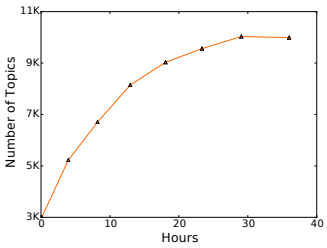


Figure 6. Growth of the tree

5. Experimental Results

Our experiments show that (1) our distributed framework achieves near-linear (i.e. near-optimal) scalability with increasing number of cores/machines; (2) the DNTs system enables big tree models (10K nodes) on large data, and well captures long-tail topics. (3) the proposed VI algorithm achieves competitive heldout likelihood with MCMC, and discovers meaningful topic evolution.

5.1. Setup

Datasets We use three public corpora for the evaluation:

- **PubMed:** 8,400,000 PubMed abstracts. The vocabulary is pruned to 70,000 words. Since no time stamp is associated, we treat the whole dataset as from 1 epoch.
- **PNAS:** 79,800 paper titles from the Proceedings of the National Academy of Sciences 1915-2005. The vocabulary size is 36,901. We grouped the titles into 10 ten-year epoches.
- **NIPS:** 1,740 documents from the Proceedings of the NIPS 1988-1999. The vocabulary size is 13,649. We grouped the documents into 12 one-year epoches.

Parameter setting For all the experiments, we set $\kappa_0 = 100, \kappa_1 = \kappa_2 = 1$, and used stick breaking parameters $\alpha = \gamma = 0.5$. We estimate each node's vMF concentration parameter β from the data according to (Banerjee et al., 2005). The staleness s for bounded-asynchronous data parallelism is set to 0, which means workers always get up-to-date global parameters from the PS.

Compute cluster All experiments were run on a compute cluster where each machine has 16 cores and 128GB RAM, connected via 1Gbps ethernet.

5.2. Scalability

We evaluate the scalability of our distributed system w.r.t the number of cores. Figure 3 shows the convergence curves on the PubMed dataset. The y-axis represents the per-document heldout likelihood (on 10% heldout test set). With increasing number of cores/machines, the model consistently converges faster (and achieves better results, possibly because the noisy parallel updates help to find a better local optimum).

To measure the speedup gain by distributed training, we record the running time for each machine setting that the heldout likelihood achieves L_1, where L_1 is the convergence likelihood of single machine (16 cores). The speedup factor is then computed as the ratio of running time for 1 machine compared to n machines. As shown in Figure 4, our distributed framework demonstrates near-linear speedup. Specifically, the speedup of using 4 machines over 1 machine achieves 3.85x.

Near-linear scalability is necessary for industrial scale applications where hundreds or thousands of machines are used to process billions of samples. Our carefully-designed data- and model-parallel schemes enable our DNTs framework to be highly scalable: the data-parallel parameter learning aggregates variational updates from different processors without need of costly tree alignment; while the model-parallel structure updating explores the tree space by highly-concurrent birth-merge operations, and aligns model structures via a communication-efficient synchronization protocol. By interleaving the two modes and keeping processors independent with each other in each mode, high degree of parallelism and scalability is achieved.

Comparison with parallel MCMC Since the parallel MCMC algorithm (Dubey et al., 2014) (§2) is too slow to converge on the PubMed dataset, we compare the speedup on the smaller PNAS data. As shown in Figure 5. Our system is significantly more scalable than MCMC which brings only 1.37x speedup on 8 cores. The reason is that, the parallel samplers maintain a shared tree model in order to avoid costly inter-processor tree alignment. This in turn leads to heavy lock contention during sampling, as the tree nodes keep being deleted and created.

Our system maintains a model copy on each processor to maximize parallelization for parameter updates, but still keeps the state alignment efficient by exchanging among processors conflict-free lightweight operation records. We compare the running time in the supplements.

5.3. Big Model for Long-tail Semantics

In the parametric literature, big models with millions of components and trillions of parameters have been shown to capture long-tail semantics from web-scale corpora, which

mass (π_ϵ)	top-6 words	topic label
27.9	at2r, at1r, receptor, type_2, angiotensin_ii, ang_ii	**Angiotensin II Type-2 Receptor**
5.9	fusion, thoracoscopic, spinal, treatment, level, surgery	**Thoracoscopic Spinal Fusion**
4.1	azt, nucleosides, pharmacokinetic, monkey, subcutaneous, methyl	**Pharmacokinetics with AZT in Monkeys**

Table 2. Long-tail topics from PubMed data. The topic labels are obtained by looking into the documents associated with the topics.

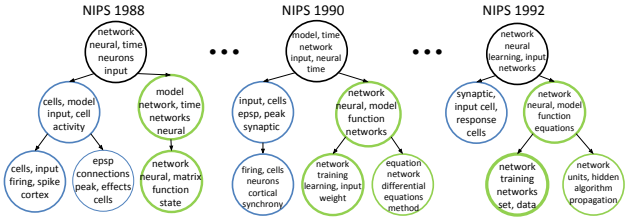


Figure 7. Variation of topic hierarchies on the NIPS data. Top-5 words of each topic are shown. The thickness of the circles represents the topic popularity. Nodes with small mass are omitted.

can improve the performance of industrial applications (Yuan et al., 2015; Wang et al., 2014). Nonparametric models, with their unbounded parameter space, provide an attractive and automatic way to capture such long tails without knowing their true size in advance, and our system facilitates the learning of truncation-free big models on large data.

As an example, Figure 6 shows that the nonparametric tree on the PubMed dataset (using 4 machines) converges to over 10K nodes organized into 8 layers. We now look at the long tail: Table 2 shows three topics with small mass, that form coherent themes corresponding to small-but-specific sets of documents. Such ability to target fine-grained topics is valuable to industrial search engines and online ad systems (Wang et al., 2014).

5.4. Quality of Learned Topics

We also evaluate the quality of the learned topics by heldout average log-likelihood. Table 3 compares with the MCMC. Our algorithm gives competitive heldout likelihood, validating that the proposed VI is a viable alternative to MCMC for discovering high-quality topics.

Method	MCMC (Dubey et al., 2014)	Our algorithm
PNAS	4562±116	4479±103
NIPS	9811±232	9712±143

Table 3. Heldout average log-likelihood on two datasets.

Our approach also discovers meaningful topic hierarchies and variation. Figure 7 shows part of the results obtained from the NIPS dataset. We can observe two major fields in the early stage of NIPS, namely, the "cell synapse"-related research and "neural network"-related research; and both the popularity and research focuses were changing along the time.

6. Related Work

There has been growing interest in scalable inference for BNPs, where much previous work has relied on data-parallelism (Smyth et al., 2009; Doshi-Velez et al., 2009). One challenge is that maintaining model parameters requires local statistics to be combined, which becomes difficult whenever local model states become misaligned (e.g. different number of clusters on each worker) (Pan et al., 2013). Thus, one must either resort to costly inter-worker model alignment, or approximation strategies that can cause slow convergence and inaccurate results (Willia-mson et al., 2013). We tackle this via an efficient alignment strategy based on conflict-free model partitioning and lightweight record transmission.

A few recent MCMC algorithms are both data- and model-parallel (Chang & Fisher III, 2014; Willia-mson et al., 2013), but their reported scales were limited to small datasets with 100s of components, and it is unclear how these methods can be efficiently applied in the distributed setting (Gal & Ghahramani, 2014). Our problem scales are orders of magnitude larger, which is necessary to handle real-world large data and capture long-tail semantics.

An alternative to MCMC for BNP inference is the variational method (Blei et al., 2006), where Bryant & Sudderth 2012 incorporated split-merge moves (Jain & Neal, 2004) to enable unbounded number of components. Our DNTs inference algorithm extends memoized VI (Hughes & Sudderth, 2013; Neal & Hinton, 1998), by applying Taylor approximation (Ahmed & Xing, 2007) to tackle intractability, and coupling it with a highly-parallelizable birth-merge scheme.

7. Conclusion

We presented a large-scale distributed system for learning dependent nonparametric trees (DNTs), which uses a novel, distributed truncation-free memoized VI algorithm that exploits both data- and model-parallelism for fast inference. Efficiency and scalability are achieved via: (a) conflict-free partitioning and birth-merge operations on each model partition; (b) synchronization of local model states via lightweight operation records. Consequently, our distributed system handles much larger problem sizes than recent BNP training frameworks. We believe that these strategies, particularly the model alignment scheme and the combined data- and model-parallelism, could be useful for scaling up other BNP models.

Acknowledgements: This research is supported by NSF IIS1218282 and NIH R01GM087694.

References

Ahmed, Amr and Xing, Eric P. On tight approximate inference of the logistic-normal topic admixture model. In *Proc. of AISTATS*, pp. 19–26, 2007.

Ahmed, Amr, Ho, Qirong, Teo, Choon H, Eisenstein, Jacob, Xing, Eric P, and Smola, Alex J. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *Proc. of AISTATS*, pp. 101–109, 2011.

Banerjee, Arindam, Dhillon, Inderjit S, Ghosh, Joydeep, and Sra, Suvrit. Clustering on the unit hypersphere using von Mises-Fisher distributions. In *JMLR*, pp. 1345–1382, 2005.

Blei, David M, Ng, Andrew Y, and Jordan, Michael I. Latent Dirichlet Allocation. *JMLR*, 3:993–1022, 2003.

Blei, David M, Jordan, Michael I, et al. Variational inference for Dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

Blei, David M, Griffiths, Thomas L, and Jordan, Michael I. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *JACM*, 57(2): 7, 2010.

Bryant, Michael and Sudderth, Erik B. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Proc. of NIPS*, pp. 2699–2707, 2012.

Chang, Jason and Fisher III, John W. Parallel sampling of HDPs using sub-cluster splits. In *Proc. of NIPS*, pp. 235–243, 2014.

Dai, Wei, Kumar, Abhimanu, Wei, Jinliang, Ho, Qirong, Gibson, Garth, and Xing, Eric P. High-performance distributed ml at scale through parameter server consistency models. In *AAAI*. 2015.

Doshi-Velez, Finale, Mohamed, Shakir, Ghahramani, Zoubin, and Knowles, David A. Large scale nonparametric bayesian inference: Data parallelisation in the indian buffet process. In *Proc. of NIPS*, pp. 1294–1302, 2009.

Dubey, Kumar, Ho, Qirong, Williamson, Sinead A, and Xing, Eric P. Dependent nonparametric trees for dynamic hierarchical clustering. In *Proc. of NIPS*, pp. 1152–1160, 2014.

Gal, Yarin and Ghahramani, Zoubin. Pitfalls in the use of parallel inference for the Dirichlet process. In *Proc. of ICML*, pp. 208–216, 2014.

Ghahramani, Zoubin, Jordan, Michael I, and Adams, Ryan P. Tree-structured stick breaking for hierarchical data. In *Proc. of NIPS*, pp. 19–27, 2010.

Ho, Qirong, Eisenstein, Jacob, and Xing, Eric P. Document hierarchies from text and links. In *Proc. of WWW*, pp. 739–748. ACM, 2012.

Ho, Qirong, Cipar, James, Cui, Henggang, Lee, Seunghak, Kim, Jin Kyu, Gibbons, Phillip B, Gibson, Garth A, Ganger, Greg, and Xing, Eric P. More effective distributed ML via a stale synchronous parallel parameter server. In *Proc. of NIPS*, pp. 1223–1231, 2013.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John. Stochastic variational inference. *JMLR*, 14(1):1303–1347, 2013.

Hughes, Michael C and Sudderth, Erik. Memoized online variational inference for Dirichlet process mixture models. In *Proc. of NIPS*, pp. 1133–1141, 2013.

Jain, Sonia and Neal, Radford M. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1), 2004.

Neal, Radford M and Hinton, Geoffrey E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer, 1998.

Pan, Xinghao, Gonzalez, Joseph E, Jegelka, Stefanie, Broderick, Tamara, and Jordan, Michael I. Optimistic concurrency control for distributed unsupervised learning. In *Proc. of NIPS*, pp. 1403–1411, 2013.

Smyth, Padhraic, Welling, Max, and Asuncion, Arthur U. Asynchronous distributed learning of topic models. In *Proc. of NIPS*, pp. 81–88, 2009.

Sudderth, Erik B and Jordan, Michael I. Shared segmentation of natural scenes using dependent Pitman-Yor processes. In *Proc. of NIPS*, pp. 1585–1592, 2009.

Wang, Yi, Zhao, Xuemin, Sun, Zhenlong, Yan, Hao, Wang, Lifeng, Jin, Zhihui, Wang, Liubin, Gao, Yang, Law, Ching, and Zeng, Jia. Peacock: Learning long-tail topic features for industrial applications. *arXiv preprint arXiv:1405.4402*, 2014.

Willia-mson, Sinead, Dubey, Avinava, and Xing, Eric. Parallel Markov chain Monte Carlo for nonparametric mixture models. In *Proc. of ICML*, pp. 98–106, 2013.

Yuan, Jinhui, Gao, Fei, Ho, Qirong, Dai, Wei, Wei, Jinliang, Zheng, Xun, Xing, Eric P, Liu, Tie-Yan, and Ma, Wei-Ying. Lightlda: Big topic models on modest compute clusters. *Proc. of WWW*, 2015.