Streaming, Distributed Variational Inference for Bayesian Nonparametrics

Authored by:

John W. Fisher III
Trevor Campbell
Jonathan P. How
Julian Straub

Abstract

This paper presents a methodology for creating streaming, distributed inference algorithms for Bayesian nonparametric (BNP) models. In the proposed framework, processing nodes receive a sequence of data minibatches, compute a variational posterior for each, and make asynchronous streaming updates to a central model. In contrast to previous algorithms, the proposed framework is truly streaming, distributed, asynchronous, learning-rate-free, and truncation-free. The key challenge in developing the framework, arising from fact that BNP models do not impose an inherent ordering on their components, is finding the correspondence between minibatch and central BNP posterior components before performing each update. To address this, the paper develops a combinatorial optimization problem over component correspondences, and provides an efficient solution technique. The paper concludes with an application of the methodology to the DP mixture model, with experimental results demonstrating its practical scalability and performance.

1 Paper Body

Bayesian nonparametric (BNP) stochastic processes are streaming priors ? their unique feature is that they specify, in a probabilistic sense, that the complexity of a latent model should grow as the amount of observed data increases. This property captures common sense in many data analysis problems ? for example, one would expect to encounter far more topics in a document corpus after reading 106 documents than after reading 10 ? and becomes crucial in settings with unbounded, persistent streams of data. While their fixed, parametric cousins can be used to infer model complexity for datasets with known magnitude a priori [1, 2], such priors are silent with respect to notions of model complexity growth in streaming data settings. Bayesian nonparametrics are also

naturally suited to parallelization of data processing, due to the exchangeability, and thus conditional independence, they often exhibit via de Finetti?s theorem. For example, labels from the Chinese Restaurant process [3] are rendered i.i.d. by conditioning on the underlying Dirichlet process (DP) random measure, and feature assignments from the Indian Buffet process [4] are rendered i.i.d. by conditioning on the underlying beta process (BP) random measure. Given these properties, one might expect there to be a wealth of inference algorithms for BNPs that address the challenges associated with parallelization and streaming. However, previous work has only addressed these two settings in concert for parametric models [5, 6], and only recently has each been addressed individually for BNPs. In the streaming setting, [7] and [8] developed streaming inference for DP mixture models using sequential variational approximation. Stochastic variational inference [9] and related methods [10?13] are often considered streaming algorithms, but their performance depends on the choice of a learning rate and on the dataset having known, fixed size a priori [5]. Outside of variational approaches, which are the focus of the present paper, there exist exact parallelized MCMC methods for BNPs [14, 15]; the tradeoff in using such methods is that they provide samples from the posterior rather than the distribution itself, and results regarding assessing 1

?m3 Minibatch Posterior
?i2 Retrieve Original
?o1
Central Posterior
Central Node
Intermediate Posterior
?o1
?i1
Node
Central Node
?3
?
?m3
?3 (= ?m2)
?m2
?i2
?2
?m2
?2
(= ?i2 + ?m3 ? ?0)
?m1
?i1
?1
?m1
?1
(= ?i1 + ?m1 ? ?o1)
Node

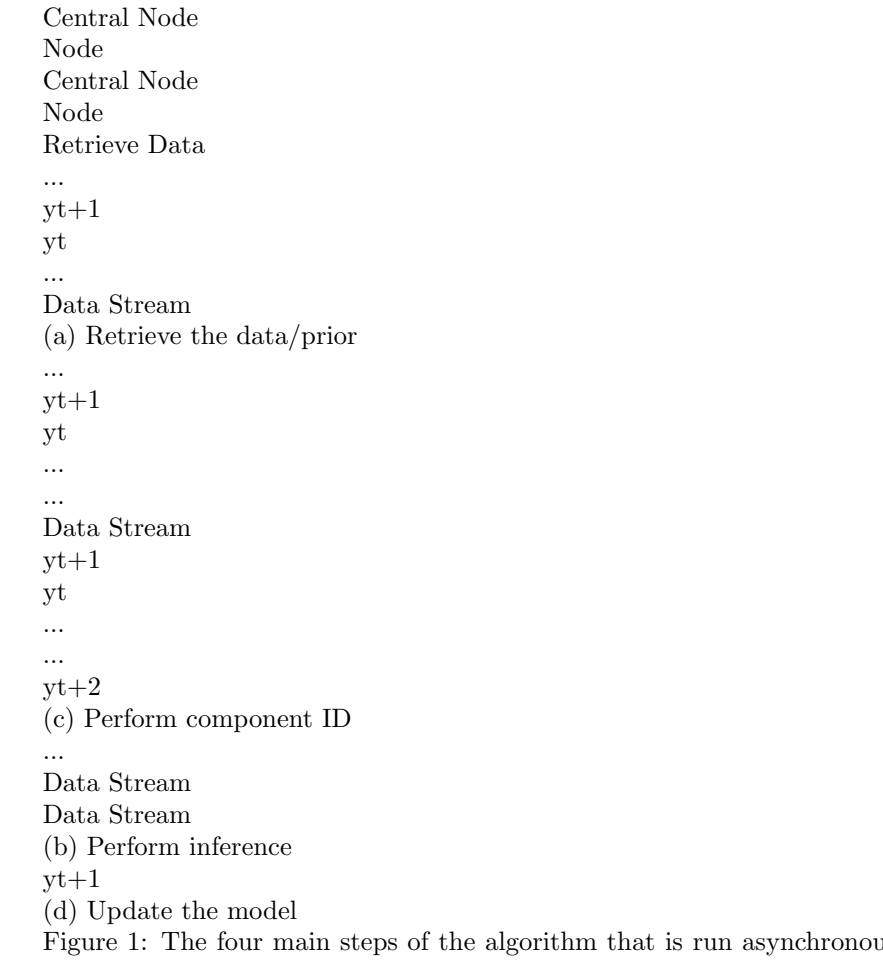

Figure 1: The four main steps of the algorithm that is run asynchronously on each processing node.

convergence remain limited. Sequential particle filters for inference have also been developed [16], but these suffer issues with particle degeneracy and exponential forgetting. The main challenge posed by the streaming, distributed setting for BNPs is the combinatorial problem of component identification. Most BNP models contain some notion of a countably infinite set of latent ?components? (e.g. clusters in a DP mixture model), and do not impose an inherent ordering on the components. Thus, in order to combine information about the components from multiple processors, the correspondence between components must first be found. Brute force search is in2 tractable even for moderately sized models ? there are K1K+K possible correspondences for two 1 sets of components of sizes K1 and K2 . Furthermore, there does not yet exist a method to evaluate the quality of a component correspondence for BNP models. This issue has been studied before in the MCMC literature, where it is known as the ?label switching problem?, but past solution techniques are generally model-specific and restricted to use on very simple mixture models [17, 18]. This

paper presents a methodology for creating streaming, distributed inference algorithms for Bayesian nonparametric models. In the proposed framework (shown for a single node A in Figure 1), processing nodes receive a sequence of data minibatches, compute a variational posterior for each, and make asynchronous streaming updates to a central model using a mapping obtained from a component identification optimization. The key contributions of this work are as follows. First, we develop a minibatch posterior decomposition that motivates a learning-rate-free streaming, distributed framework suitable for Bayesian nonparametrics. Then, we derive the component identification optimization problem by maximizing the probability of a component matching. We show that the BNP prior regularizes model complexity in the optimization; an interesting side effect of this is that regardless of whether the minibatch variational inference scheme is truncated, the proposed algorithm is truncation-free. Finally, we provide an efficiently computable regularization bound for the Dirichlet process prior based on Jensen?s inequality1 . The paper concludes with applications of the methodology to the DP mixture model, with experimental results demonstrating the scalability and performance of the method in practice.

2

Streaming, distributed Bayesian nonparametric inference

The proposed framework, motivated by a posterior decomposition that will be discussed in Section 2.1, involves a collection of processing nodes with asynchronous access to a central variational posterior approximation (shown for a single node in Figure 1). Data is provided to each processing node as a sequence of minibatches. When a processing node receives a minibatch of data, it obtains the central posterior (Figure 1a), and using it as a prior, computes a minibatch variational posterior approximation (Figure 1b). When minibatch inference is complete, the node then performs component identification between the minibatch posterior and the current central posterior, accounting for possible modifications made by other processing nodes (Figure 1c). Finally, it merges the minibatch posterior into the central variational posterior (Figure 1d). In the following sections, we use the DP mixture [3] as a guiding example for the technical development of the inference framework. However, it is emphasized that the material in this paper generalizes to many other BNP models, such as the hierarchical DP (HDP) topic model [19], BP latent feature model [20], and Pitman-Yor (PY) mixture [21] (see the supplement for further details). 1

Regularization bounds for other popular BNP priors may be found in the supplement.

2

2.1

Posterior decomposition

Consider a DP mixture model [3], with cluster parameters ?, assignments z, and observed data y. For each asynchronous update made by each processing node, the dataset is split into three subsets y = yo ? yi ? ym for analysis. When the processing node receives a minibatch of data ym , it queries the central processing node for the original posterior p(?, zo —yo), which will be used as the prior for minibatch inference. Once inference is complete, it

again queries the central processing node for the intermediate posterior p(?, zo , zi —yo , yi) which accounts for asynchronous updates from other processing nodes since minibatch inference began. Each subset yr , r ? {o, i, m}, has Nr r observations {yrj }N j=1 , and each variable zrj ? N assigns yrj to cluster parameter ?zrj . Given the independence of ? and z in the prior, and the conditional independence of the data given the latent parameters, Bayes? rule yields the following decomposition of the posterior of ? and z given y, Updated Central Posterior

z }— { p(?, z—y) ?

Original Posterior

Minibatch Posterior

Intermediate Posterior

z }— { z }— { z }— { p(zi , zm —zo) ? p(?, zo —yo)?1 ? p(?, zm , zo —ym , yo) ? p(?, zi , zo —yi , yo). p(zi —zo)p(zm —zo)

(1)

This decomposition suggests a simple streaming, distributed, asynchronous update rule for a processing node: first, obtain the current central posterior density p(?, zo —yo), and using it as a prior, compute the minibatch posterior p(?, zm , zo —yo , ym); and then update the central posterior density by using (1) with the current central posterior density p(?, zi , zo —yi , yo). However, there are two issues preventing the direct application of the decomposition rule (1): Unknown component correspondence: Since it is generally intractable to find the minibatch posteriors p(?, zm , zo —yo , ym) exactly, approximate methods are required. Further, as (1) requires the multiplication of densities, sampling-based methods are difficult to use, suggesting a variational approach. Typical mean-field variational techniques introduce an artificial ordering of the parameters in the posterior, thereby breaking symmetry that is crucial to combining posteriors correctly using density multiplication [6]. The use of (1) with mean-field variational approximations thus requires first solving a component identification problem. Unknown model size: While previous posterior merging procedures required a 1-to-1 matching between the components of the minibatch posterior and central posterior [5, 6], Bayesian nonparametric posteriors break this assumption. Indeed, the datasets yo , yi , and ym from the same nonparametric mixture model can be generated by the same, disjoint, or an overlapping set of cluster parameters. In other words, the global number of unique posterior components cannot be determined until the component identification problem is solved and the minibatch posterior is merged. 2.2

Variational component identification

Suppose we have the following mean-field exponential family prior and approximate variational posterior densities in the minibatch decomposition (1),

T

p(?k) = h(?k)e?0 T (?k)?A(?0) ?k ? N p(?, zo —yo) ' qo (?, zo) = ?o (zo)

Ko Y

T

h(?k)e?ok T (?k)?A(?ok)

k=1
p(?, zm , zo —ym , yo) ’ qm (?, zm , zo) = ?m (zm)?o (zo)
K m Y
T
h(?k)e?mk T (?k)?A(?mk)
(2)
k=1
p(?, zi , zo —yi , yo) ’ qi (?, zi , zo) = ?i (zi)?o (zo)
Ki Y
T
h(?k)e?ik T (?k)?A(?ik) ,
k=1

where ?r (?), r ? {o, i, m} are products of categorical distributions for the cluster labels zr , and the goal is to use the posterior decomposition (1) to find the updated posterior approximation K Y
p(?, z—y) ’ q(?, z) = ?(z)
k=1
3
T
h(?k)e?k T (?k)?A(?k) .
(3)

As mentioned in the previous section, the artificial ordering of components causes the na??ve application of (1) with variational approximations to fail, as disparate components from the approximate posteriors may be merged erroneously. This is demonstrated in Figure 3a, which shows results from a synthetic experiment (described in Section 4) ignoring component identification. As the number of parallel threads increases, more matching mistakes are made, leading to decreasing model quality. To address this, first note that there is no issue with the first Ko components of qm and qi ; these can be merged directly since they each correspond to the Ko components of qo . Thus, the component 0 identification problem reduces to finding the correspondence between the last Km = Km ? Ko 0 components of the minibatch posterior and the last Ki = Ki ? Ko components of the intermediate posterior. For notational simplicity (and without loss of generality), fix the component ordering 0 of the intermediate posterior qi , and define ? : [Km] ? [Ki + Km] to be the 1-to-1 mapping from minibatch posterior component k to updated central posterior component ?(k), where [K] := {1, . . . , K}. The fact that the first Ko components have no ordering ambiguity can be expressed as 0 ?(k) = k ?k ? [Ko]. Note that the maximum number of components after merging is Ki + Km , 0 since each of the last Km components in the minibatch posterior may correspond to new components in the intermediate posterior. After substituting the three variational approximations (2) into (1), the goal of the component identification optimization is to find the 1-to-1 mapping ? ? that yields the largest updated posterior normalizing constant, i.e. matches components with similar densities, XZ p(zi , zm —zo) ? ? ? ? argmax qo (?, zo)?1 qm (?, zm , zo)qi (?, zi , zo) p(z —z)p(z —z) ? i o m o ? z s.t.

? ? qm (?, zm) = ?m (zm)
K m Y
T
h(??(k))e?mk T (??(k))?A(?mk)
(4)
k=1

?(k) = k, ?k ? [Ko] , ? 1-to-1 ? ? (zmj = ?(k)) = P? (zmj = k). (zm) is the distribution such that P?m Taking the where ?m m logarithm of the objective and exploiting the mean-field decoupling allows the separation of the objective into a sum of two terms: one expressing the quality of the matching between components (the integral over ?), and one that regularizes the final model size (the sum over z). While the first term is available in closed form, the second is in general not. Therefore, using the concavity of the logarithm function, Jensen?s inequality yields a lower bound that can be used in place of the intractable original objective, resulting in the final component identification optimization: 0 Ki +Km

?
? ? argmax ?
X
A (? ?k?) + E?? [log p(zi , zm , zo)] (5)
k=1
s.t.
? ??k? = ??ik + ??mk ? ??ok ?(k) = k ?k ? [Ko] , ? 1-to-1.

A more detailed derivation of the optimization may be found in the supplement. E?? denotes expec? tation under the distribution ?o (zo)?i (zi)?m (zm), and

?m??1 (k) k ? ? ([Km]) ?rk k ? Kr ? ??rk = ?r ? {o, i, m}, ??mk = , (6)
?0 k ¿ Kr ?0 k? / ? ([Km]) where ? ([Km]) denotes the range of the mapping ?. The definitions in (6) ensure that the prior ?0 is used whenever a posterior r ? {i, m, o} does not contain a particular component k. The intuition for the optimization (5) is that it combines finding component correspondences with high similarity (via the log-partition function) with a regularization term2 on the final updated posterior model size. Despite its motivation from the Dirichlet process mixture, the component identification optimization (5) is not specific to this model. Indeed, the derivation did not rely on any properties specific to the Dirichlet process mixture; the optimization applies to any Bayesian nonparametric model with a set of ?components? ?, and a set of combinatorial ?indicators? z. For example, the optimization applies to the hierarchical Dirichlet process topic model [10] with topic word distributions ? and local-toglobal topic correspondences z, and to the beta process latent feature model [4] with features ? and 2

h i
? This is equivalent to the KL-divergence regularization ?KL ?o (zo)?i (zi)?m (zm) p(zi , zm , zo) .

4

binary assignment vectors z. The form of the objective in the component identification optimization (5) reflects this generality. In order to apply the proposed streaming, distributed method to a particular model, one simply needs a black-box variational inference algorithm that computes posteriors of the form (2), and a way to compute or bound the expectation in the objective of (5). 2.3

Updating the central posterior

To update the central posterior, the node first locks it and solves for ? ? via (5). Locking prevents other nodes from solving (5) or modifying the central posterior, but does not prevent other nodes from reading the central posterior, obtaining minibatches, or performing inference; the synthetic experiment in Section 4 shows that this does not incur a significant time penalty in practice. Then the processing node transmits ? ? and its minibatch variational posterior to the central processing node where the product decomposition (1) is used to find the updated central variational posterior q in (3), with parameters

?? ?? K = max Ki , max ? ? (k) , ?(z) = ?i (zi)?o (zo)?m (zm), ?k = ??ik + ??mk ? ??ok . (7) k?[Km]

Finally, the node unlocks the central posterior, and the next processing node to receive a new minibatch will use the above K, ?(z), and ?k from the central node as their Ko , ?o (zo), and ?ok .

3

Application to the Dirichlet process mixture model

The expectation in the objective of (5) is typically intractable to compute in closed-form; therefore, a suitable lower bound may be used in its place. This section presents such a bound for the Dirichlet process, and discusses the application of the proposed inference framework to the Dirichlet process mixture model using the developed bound. Crucially, the lower bound decomposes such that the optimization (5) becomes a maximum-weight bipartite matching problem. Such problems are solvable in polynomial time [22] by the Hungarian algorithm, leading to a tractable component identification step in the proposed streaming, distributed framework. 3.1

Regularization lower bound

For the Dirichlet process with concentration parameter ? ¿ 0, p(zi , zm , zo) is the Exchangeable Partition Probability Function (EPPF) [23] Y p(zi , zm , zo) ? ?—K—?1 (nk ? 1)!, (8) k?K

where nk is the amount of data assigned to cluster k, and K is the set of labels of nonempty clusters. Given that the variational distribution ?r (zr), r ? {i, m, o} is a product of independent categorQNr QKr 1[zrj =k] ical distributions ?r (zr) = j=1 , Jensen?s inequality may be used to bound the k=1 ?rjk regularization in (5) below (see the supplement for further details) by 0 Ki +Km

E?? [log p(zi , zm , zo)] ?

X

? 1 ? es?k log ? + log ? max 2, t??k + C

k=1

s??k = s?ik + s??mk + s?ok ,

(9)

t??k = t?ik + t??mk + t?ok ,

where C is a constant with respect to the component mapping ?, and PN PN r r k?Kr k?Kr j=1 log(1??rjk) j=1 ?rjk s?rk = ?r?{o,i,m} t?rk = ?r?{o,i,m} 0 k¿Kr 0 k¿Kr PN PN (10) m m k??([Km]) k??([Km]) j=1 log(1??mj? ?1 (k)) j=1 ?mj? ?1 (k) s??mk = t??mk = . 0

k??([K / m])

k??([K / m])

0

Note that the bound (9) allows incremental updates: after finding the optimal mapping ? ? , the central update (7) can be augmented by updating the values of sk and tk on the central node to ?

?

sk ? s?ik + s??mk + s?ok ,

tk ? t?ik + t??mk + t?ok . 5

(11)

600

140 Truth Lower Bound

Truth Lower Bound 120

200

Regularization

Regularization

400

Increasing ?

0

-200

Increasing ? 80

60

-400

-600

100

0

20

40

60

80

40 0.001 Certain

100

Number of Clusters

0.01

(a)

0.1

Clustering Uncertainty ?

1.0

1000 Uncertain

(b)

Figure 2: The Dirichlet process regularization and lower bound, with (2a) fully uncertain labelling and varying number of clusters, and (2b) the number of clusters fixed with varying labelling uncertainty.

As with K, ?k , and ? from (7), after performing the regularization statistics update (11), a processing node that receives a new minibatch will use the above sk and tk as their sok and tok , respectively. Figure 2 demonstrates the behavior of the lower bound in a synthetic experiment with N = 100

datapoints for various DP concentration parameter values ? ? 10?3 , 103 . The true regularization log E? [p(z)] was computed by sample approximation with 104 samples. In Figure 2a, the number of 1 . This figure demonstrates clusters K was varied, with symmetric categorical label weights set to K two important phenomena. First, the bound increases as K ? 0; in other words, it gives preference to fewer, larger clusters, which is the typical BNP ?rich get richer? property. Second, the behavior of the bound as K ? N depends on the concentration parameter ? ? as ? increases, more clusters are preferred. In Figure 2b, the number of clusters K was fixed to 10, and the categorical

label weights were sampled from a symmetric Dirichlet distribution with parameter ? ? 10?3 , 103 . This figure demonstrates that the bound does not degrade significantly with high labelling uncertainty, and is nearly exact for low labelling uncertainty. Overall, Figure 2a demonstrates that the proposed lower bound exhibits similar behaviors to the true regularization, supporting its use in the optimization (5). 3.2

Solving the component identification optimization

Given that both the regularization (9) and component matching score in the objective (5) decompose 0], the objective can be rewritten using a matrix of matching as a sum of terms for each k ? [Ki + Km 0 0 0 0 Ki +Km ?(Ki +Km ( )) scores R ? R and selector variables X ? {0, 1}(Ki +Km)?(Ki +Km) . Setting Xkj = 1 indicates that component k in the minibatch posterior is matched to component j in the intermediate posterior (i.e. ?(k) = j), providing a score Rkj defined using (6) and (10) as

Rkj = A (? ?ij + ??mk ? ??oj)+ 1 ? es?ij +?smk +?soj log ?+log ? max 2, t?ij + t?mk + t?oj . (12) The optimization (5) can be rewritten in terms of X and R as

X? ? argmax tr XT R X

XT 1 = 1, Xkk = 1, ?k ? [Ko] 0 0 T X ? {0, 1}(Ki +Km)?(Ki +Km) , 1 = [1, . . . , 1] .

s.t. X1 = 1,

(13)

The first two constraints express the 1-to-1 property of ?(?). The constraint Xkk = 1?k ? [Ko] fixes the upper Ko ?Ko block of X to I (due to the fact that the first Ko components are matched directly), 0 0 and the off-diagonal blocks to 0. Denoting X0 , R0 to be the lower right (Ki0 + Km) ? (Ki0 + Km) 0 blocks of X, R, the remaining optimization problem is a linear assignment problem on X with cost matrix ?R0 , which can be solved using the Hungarian algorithm3 . Note that if Km = Ko or Ki = Ko , this implies that no matching problem needs to be solved ? the first Ko components 0 of the minibatch posterior are

matched directly, and the last Km are set as new components. In practical implementation of the framework, new clusters are typically discovered at a diminishing rate as more data are observed, so the number of matching problems that are solved likewise tapers off. The final optimal component mapping ? ? is found by finding the nonzero elements of X? : ? ? (k) ? argmax X?kj ?k ? [Km] . j

3

For the experiments in this work, we used the implementation at github.com/hrldcpr/hungarian.

6

(14)

CPU Time (s)

-5.0

12.0

-6.0

10.0

-7.0

6.0

-8.0

4.0

-9.0

-10.0

2.0

-10.0

-11.0

0.0

-7.0

6.0

-8.0

4.0

-9.0

2.0

0.0

1

2

4

8

16

24

32

40

48

1

2

4

8

16

24
32
40
48
SDA-DP Batch SVA SVI moVB SC
-7
-8
-9
-10
-11
-11.0
-5
-4
-3
-2
-1
(a) SDA-DP without component ID
(b) SDA-DP with component ID
45
Clusters # Matchings True # Clusters
120
20
3
4
100
100
80
80
60
Clusters # Matchings True # Clusters
120
Count
Count
25
2
140
35 30
1
(c) Test log-likelihood traces
140
40
0
Time (Log10 s)
Threads
Threads
Count

-6.0
8.0
8.0
-6
-5.0 CPU Time Test Log Likelihood
Test Log Likelihood Test Log Likelihood
CPU Time Test Log Likelihood
10.0
Test Log Likelihood CPU Time (s)
12.0
60
15
40
40
10
20
5 0
0
10
20
30
40
50
60
Merge Time (microseconds)
(d) CPU time for component ID
70
0
20
0
20
40
60
80
Minibatches Merged
(e) Cluster/component ID counts
100
0
1
2
4
8
16
24
32
40

48

Threads

(f) Final cluster/component ID counts

Figure 3: Synthetic results over 30 trials. (3a-3b) Computation time and test log likelihood for SDA-DP with varying numbers of parallel threads, with component identification disabled (3a) and enabled (3b). (3c) Test log likelihood traces for SDA-DP (40 threads) and the comparison algorithms. (3d) Histogram of computation time (in microseconds) to solve the component identification optimization. (3e) Number of clusters and number of component identification problems solved as a function of the number of minibatch updates (40 threads). (3f) Final number of clusters and matchings solved with varying numbers of parallel threads.

4

Experiments

In this section, the proposed inference framework is evaluated on the DP Gaussian mixture with a normal-inverse-Wishart (NIW) prior. We compare the streaming, distributed procedure coupled with standard variational inference [24] (SDA-DP) to five state-of-the-art inference algorithms: memoized online variational inference (moVB) [13], stochastic online variational inference (SVI) [9] with 1 learning rate (t+10)? 2 , sequential variational approximation (SVA) [7] with cluster creation thresh?1 old 10 and prune/merge threshold 10?3 , subcluster splits MCMC (SC) [14], and batch variational inference (Batch) [24]. Priors were set by hand and all methods were initialized randomly. Methods that use multiple passes through the data (e.g. moVB, SVI) were allowed to do so. moVB was allowed to make birth/death moves, while SVI/Batch had fixed truncations. All experiments were performed on a computer with 24 CPU cores and 12GiB of RAM. Synthetic: This dataset consisted of 100,000 2-dimensional vectors generated from a Gaussian mixture model with 100 clusters and a NIW(?0 , ?0 , ?0 , ?0) prior with ?0 = 0, ?0 = 10?3 , ?0 = I, and ?0 = 4. The algorithms were given the true NIW prior, DP concentration ? = 5, and minibatches of size 50. SDA-DP minibatch inference was truncated to K = 50 components, and all other algorithms were truncated to K = 200 components. Figure 3 shows the results from the experiment over 30 trials, which illustrate a number of important properties of SDA-DP. First and foremost, ignoring the component identification problem leads to decreasing model quality with increasing number of parallel threads, since more matching mistakes are made (Figure 3a). Second, if component identification is properly accounted for using the proposed optimization, increasing the number of parallel threads reduces execution time, but does not affect the final model quality (Figure 3b). Third, SDA-DP (with 40 threads) converges to the same final test log likelihood as the comparison algorithms in significantly reduced time (Figure 3c). Fourth, each component identification optimization typically takes ? 10?5 seconds, and thus matching accounts for less than a millisecond of total computation and does not affect the overall computation time significantly (Figure 3d). Fifth, the majority of the component matching problems are solved within the first 80 minibatch updates (out of a total of 2,000) ? afterwards, the true clusters have

all been discovered and the processing nodes contribute to those clusters rather than creating new ones, as per the discussion at the end of Section 3.2 (Figure 3e). Finally, increased parallelization can be advantageous in discovering the correct number of clusters; with only one thread, mistakes made early on are built upon and persist, whereas with more threads there are more component identification problems solved, and thus more chances to discover the correct clusters (Figure 3f). 7

3500 3000
Count
2500 2000 1500 1000 500 0
0
1
2
3
4
5
6
7
8
9
Cluster

(a) Airplane trajectory clusters

Algorithm SDA-DP SVI SVA moVB SC Batch

(b) Airplane cluster weights

(c) MNIST clusters

(d) Numerical results on Airplane, MNIST, and SUN Airplane MNIST SUN Time (s) TestLL Time (s) TestLL Time (s) TestLL 0.66 -0.55 3.0 -145.3 9.4 -150.3 1.50 -0.59 117.4 -147.1 568.9 -149.9 3.00 -4.71 57.0 -145.0 10.4 -152.8 0.69 -0.72 645.9 -149.2 1258.1 -149.7 393.6 -1.06 1639.1 -146.8 1618.4 -150.6 1.07 -0.72 829.6 -149.5 1881.5 -149.7

Figure 4: (4a-4b) Highest-probability instances and counts for 10 trajectory clusters generated by SDA-DP. (4c) Highest-probability instances for 20 clusters discovered by SDA-DP on MNIST. (4d) Numerical results.

Airplane Trajectories: This dataset consisted of ?3,000,000 automatic dependent surveillance broadcast (ADS-B) messages collected from planes across the United States during the period 201303-22 01:30:00UTC to 2013-03-28 12:00:00UTC. The messages were connected based on plane call sign and time stamp, and erroneous trajectories were filtered based on reasonable spatial/temporal bounds, yielding 15,022 trajectories with 1,000 held out for testing. The latitude/longitude points in each trajectory were fit via linear regression, and the 3-dimensional parameter vectors were clustered. Data was split into minibatches of size 100, and SDA-DP used 16 parallel threads. MNIST Digits [25]: This dataset consisted of 70,000 28 ? 28 images of hand-written digits, with 10,000 held out for testing. The images were reduced to 20 dimensions with PCA prior to clustering. Data was split into minibatches of size 500, and SDA-DP used 48 parallel threads. SUN Images [26]: This dataset consisted of 108,755 images

from 397 scene categories, with 8,755 held out for testing. The images were reduced to 20 dimensions with PCA prior to clustering. Data was split into minibatches of size 500, and SDA-DP used 48 parallel threads. Figure 4 shows the results from the experiments on the three real datasets. From a qualitative standpoint, SDA-DP discovers sensible clusters in the data, as demonstrated in Figures 4a?4c. However, an important quantitative result is highlighted by Table 4d: the larger a dataset is, the more the benefits of parallelism provided by SDA-DP become apparent. SDA-DP consistently provides a model quality that is competitive with the other algorithms, but requires orders of magnitude less computation time, corroborating similar findings on the synthetic dataset.

5

Conclusions

This paper presented a streaming, distributed, asynchronous inference algorithm for Bayesian nonparametric models, with a focus on the combinatorial problem of matching minibatch posterior components to central posterior components during asynchronous updates. The main contributions are a component identification optimization based on a minibatch posterior decomposition, a tractable bound on the objective for the Dirichlet process mixture, and experiments demonstrating the performance of the methodology on large-scale datasets. While the present work focused on the DP mixture as a guiding example, it is not limited to this model ? exploring the application of the proposed methodology to other BNP models is a potential area for future research. Acknowledgments This work was supported by the Office of Naval Research under ONR MURI grant N000141110688.

8

2 References

[1] Agostino Nobile. Bayesian Analysis of Finite Mixture Distributions. PhD thesis, Carnegie Mellon University, 1994. [2] Jeffrey W. Miller and Matthew T. Harrison. A simple example of Dirichlet process mixture inconsistency for the number of components. In Advances in Neural Information Processing Systems 26, 2013. [3] Yee Whye Teh. Dirichlet processes. In Encyclopedia of Machine Learning. Springer, New York, 2010. [4] Thomas L. Griffiths and Zoubin Ghahramani. Infinite latent feature models and the Indian buffet process. In Advances in Neural Information Processing Systems 22, 2005. [5] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. Streaming variational Bayes. In Advances in Neural Information Procesing Systems 26, 2013. [6] Trevor Campbell and Jonathan P. How. Approximate decentralized Bayesian inference. In Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, 2014. [7] Dahua Lin. Online learning of nonparametric mixture models via sequential variational approximation. In Advances in Neural Information Processing Systems 26, 2013. [8] Xiaole Zhang, David J. Nott, Christopher Yau, and Ajay Jasra. A sequential algorithm for fast fitting of Dirichlet process mixture models. Journal of Computational and Graphical

Statistics, 23(4):1143?1162, 2014. [9] Matt Hoffman, David Blei, Chong Wang, and John Paisley. Stochastic variational inference. Journal of Machine Learning Research, 14:1303?1347, 2013. [10] Chong Wang, John Paisley, and David M. Blei. Online variational inference for the hierarchical Dirichlet process. In Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, 2011. [11] Michael Bryant and Erik Sudderth. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In Advances in Neural Information Proecssing Systems 23, 2009. [12] Chong Wang and David Blei. Truncation-free stochastic variational inference for Bayesian nonparametric models. In Advances in Neural Information Processing Systems 25, 2012. [13] Michael Hughes and Erik Sudderth. Memoized online variational inference for Dirichlet process mixture models. In Advances in Neural Information Processing Systems 26, 2013. [14] Jason Chang and John Fisher III. Parallel sampling of DP mixture models using sub-clusters splits. In Advances in Neural Information Procesing Systems 26, 2013. [15] Willie Neiswanger, Chong Wang, and Eric P. Xing. Asymptotically exact, embarassingly parallel MCMC. In Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, 2014. [16] Carlos M. Carvalho, Hedibert F. Lopes, Nicholas G. Polson, and Matt A. Taddy. Particle learning for general mixtures. Bayesian Analysis, 5(4):709?740, 2010. [17] Matthew Stephens. Dealing with label switching in mixture models. Journal of the Royal Statistical Society: Series B, 62(4):795?809, 2000. [18] Ajay Jasra, Chris Holmes, and David Stephens. Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. Statistical Science, 20(1):50?67, 2005. [19] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. Journal of the American Statistical Association, 101(476):1566?1581, 2006. [20] Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the Indian buffet process. In Proceedings of the International Conference on Machine Learning, 2009. [21] Avinava Dubey, Sinead Williamson, and Eric Xing. Parallel Markov chain Monte Carlo for Pitman-Yor mixture models. In Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, 2014. [22] Jack Edmonds and Richard Karp. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the Association for Computing Machinery, 19:248?264, 1972. [23] Jim Pitman. Exchangeable and partially exchangeable random partitions. Probability Theory and Related Fields, 102(2):145?158, 1995. [24] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. Bayesian Analysis, 1(1):121?144, 2006. [25] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST database of handwritten digits. Online: yann.lecun.com/exdb/mnist. [26] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN 397 image database. Online: vision.cs.princeton.edu/projects/2010/SUN.

9