Forgetting Counts : Constant Memory Inference for a Dependent Hierarchical Pitman-Yor Process

Nicholas Bartlett[⋆] BARTLETT@STAT.COLUMBIA.EDU
David Pfau[†] PFAU@NEUROTHEORY.COLUMBIA.EDU
Frank Wood[⋆] FWOOD@STAT.COLUMBIA.EDU

[⋆]Department of Statistics
[†]Center for Theoretical Neuroscience
Columbia University, 2960 Broadway, New York, NY 10027, USA

Abstract

We propose a novel *dependent* hierarchical Pitman-Yor process model for discrete data. An incremental Monte Carlo inference procedure for this model is developed. We show that inference in this model can be performed in *constant space* and linear time. The model is demonstrated in a discrete sequence prediction task where it is shown to achieve state of the art sequence prediction performance while using significantly less memory.

1. Introduction

In this paper we define a dependent hierarchical Pitman-Yor process (HPYP) (Teh, 2006) and develop a constant space, linear time incremental inference procedure for models of discrete data based on it. This contribution can be described as a "forgetting" procedure for existing HPYP inference procedures that retains only a "good," constant-sized subset of the training data. In designing and justifying such a model and inference procedure we extend definitions and inference methods for dependent sequences of Pitman-Yor (PY) processes (Caron et al., 2007a;b) to the hierarchical case. We propose an incremental inference procedure for the resulting model that has two interpretations: either a valid inference procedure for a sequence of dependent HPYP's or as invalid (but approximately correct) inference procedure for a single non-varying HPYP. In the latter case, our approach can be described as restricting the estimated model at all times to the constant size model that "best" approximates (in some sense of the word best) a full model that would otherwise grow in the size of the training data.

Previous work on dependent Dirichlet and Pitman-Yor processes (MacEachern, 2000; Srebro & Roweis, 2005; Griffin & Steel, 2006; Caron et al., 2007a;b) was motivated by the desire to construct generative procedures that induce dependence between related processes. In our work we are motivated instead by the goal of performing constant space inference in a HPYP. As the HPYP is a (Bayesian) nonparametric model, in order to achieve this we must "forget" data. Conveniently, forgetting is one of the main mechanisms for generating (and performing inference) in models of sequences of dependent processes. This hints at why our inference procedure can be interpreted in two ways. If the true generative process varies in some dependent way, then it is possible that the implicit dependency induced by a forgetting procedure can capture and model that variation. If, on the other hand, the true generative process does not vary, the constant memory constraint still requires that we forget. This means choosing an informative subset of the data to retain at the cost of using an estimator that may be inconsistent. This strategy of selecting a most informative subset of training data is not unique in the Bayesian nonparametric modeling literature. In sparse Gaussian process modeling, forgetting strategies are used to achieve constant time and space (independent of the number of observations) inference procedures (Lawrence et al., 2003; Csató & Opper., 2002; Snelson & Ghahramani, 2006).

Existing HPYP models have shown excellent predictive performance but are impractical to estimate for large datasets, motivating our exploration of constant space inference. A linear space, linear time incremental estimator for the sequence memoizer (SM) (Wood et al., 2009) (an HPYP model of unbounded depth for

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

discrete sequence data) has been developed (Gasthaus et al., 2010). A consequence of this development is that a SM could be deployed as the probabilistic sequence prediction model in a general purpose lossless compressor. A compressor built in this way was shown to be better than other general purpose lossless compressors (including gzip, bzip2, etc.). Unfortunately an $O(n)$ memory complexity bound (where here n is the length of the sequence) makes the SM impractical for use in a compressor. Using the SM, arbitrarily long data streams cannot be modeled on a fixed computer and therefore cannot be compressed. The obvious approach of estimating a new model for each of a number of constant size subsequences achieves the constant memory bound but, as we will show, can reliably be improved upon. We show that our proposed constant space HPYP model and inference procedure achieves near optimal results for reasonable (practical to implement on modern computers) memory constraints. This suggests that it might be possible to build and deploy an improved, practical, state-of-the-art compressor based on a constant space HPYP sequence prediction engine.

In the next section we review the Pitman-Yor process, the HPYP, and the SM. Section 3 describes dependent Pitman-Yor processes and defines a dependent hierarchical Pitman-Yor process. Section 4 describes a sequential Monte Carlo inference procedure for the dependent HPYP defined in Section 3. Finally Section 5 compares the predictive performance of the constant memory model to more complex models whose complexity is allowed to grow. Why the predictive performance of the constant memory model is comparable to that of more complex models for reasonable memory bounds is discussed in Section 6.

2. Background

2.1. Pitman-Yor Process

The Pitman-Yor process is a generalization of the Dirichlet process with three parameters. If $\mathcal{G} \sim \mathcal{PY}(d, c, \mathcal{G}_0)$ we say $\mathcal{G}$ is distributed according to a Pitman-Yor process with discount parameter d, concentration parameter c, and base measure $\mathcal{G}_0$. In the case that $d = 0$ the Pitman-Yor process reduces to the Dirichlet process (Pitman & Yor, 1997). A simple model using the Pitman-Yor process, where a distribution is drawn from a Pitman-Yor process and then samples are drawn from the resulting distribution, can be written as follows:

$$\begin{aligned} \mathcal{G}|d, c, \mathcal{G}_0 &\sim \mathcal{PY}(d, c, \mathcal{G}_0) \\ \theta_i|\mathcal{G} &\sim \mathcal{G}, \quad i = 1, \ldots, N \end{aligned}$$

It is possible to work in a representation where the random distribution $\mathcal{G}$ is analytically marginalized out. One can draw samples $\{\theta_j\}_{j=1}^N$ in this representation using a two step process. The first step produces a partition of the first N integers which follows the two parameter Ewen's sampling distribution $\mathcal{ES}_N(d, c)$ (Ewens & Tavare, 1995). The second step assigns to each of the K segments of the partition a parameter ψ_k drawn independently from $\mathcal{G}_0$. We set $\theta_j = \psi_k$ for all integers j in segment k of the partition (Blackwell & MacQueen, 1973).

Samples can be drawn from the Ewen's sampling distribution using the Chinese Restaurant Process (CRP). Since the connection between the CRP and the Ewen's sampling distribution is essential to developments later in the paper we describe here the CRP in detail. We imagine seating customers in a restaurant with an infinite number of tables. The first customer sits down at an empty table. Customers $2, \ldots, N$ are seated sequentially by seating the j^{th} customer at a table drawn from the following distribution:

$$\begin{aligned} p(\text{occupied table } i|\text{ previous}) &= \tfrac{n_i - d}{j - 1 + c} \\ p(\text{unoccupied table}|\text{ previous}) &= \tfrac{t_{j-1}d + c}{j - 1 + c} \end{aligned}$$

where n_i is the number of customers already sitting at table i, t_{j-1} is the number of tables occupied by the first $j-1$ customers, and previous refers to the seating arrangement of the first $j-1$ customers. The final seating arrangement in the restaurant defines a partition of the first N integers which follows the $\mathcal{ES}_N(d, c)$ distribution (Pitman, 1995). To complete the process of generating a sample $\{\theta_j\}_{j=1}^N$ we must endow each occupied table in the restaurant with a parameter ψ_k drawn independently from $\mathcal{G}_0$. We set $\theta_j = \psi_k$ if customer j is sitting at table k.

2.2. Hierarchical Pitman-Yor Process

An example hierarchical Pitman-Yor process is

$$\begin{aligned} \mathcal{G}_1|d_1, c_1, \mathcal{G}_0 &\sim \mathcal{PY}(d_1, c_1, \mathcal{G}_0) \\ \mathcal{G}_2|d_2, c_2, \mathcal{G}_1 &\sim \mathcal{PY}(d_2, c_2, \mathcal{G}_1) \\ \theta_i|\mathcal{G}_2 &\sim \mathcal{G}_2 \quad i = 1, \ldots, N. \end{aligned}$$

To obtain a sample from this hierarchical Pitman-Yor process it is again possible to work in a representation where $\mathcal{G}_2$ and $\mathcal{G}_1$ are analytically marginalized out. One can generate samples in this representation by recursively applying the algorithm for the single Pitman-Yor process. To draw a sample $\{\theta_j\}_{j=1}^N$ we again need to produce a partition of the first N integers following the $\mathcal{ES}_N(d_2, c_2)$ distribution. This can be achieved using the CRP. In Figure 1 this is represented by the restaurant corresponding to $\mathcal{G}_2$. We will

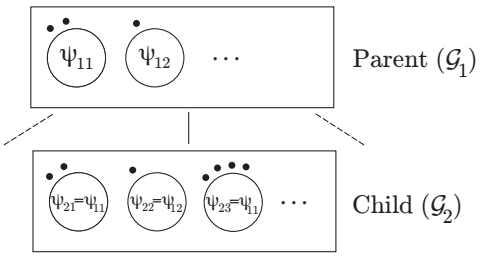

Figure 1. Chinese restaurant franchise

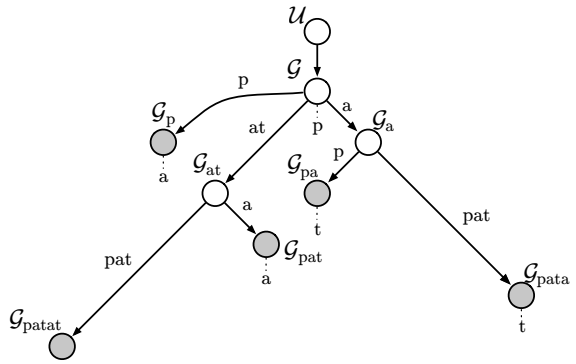

Figure 2. Sequence Memoizer graphical model

denote the number of tables in the child restaurant as K_2. Each of the K_2 tables must be endowed with a parameter ψ_{2k} drawn independently from $\mathcal{G}_1$. Since $\mathcal{G}_1$ has been marginalized out we obtain $\{\psi_{2k}\}_{k=1}^{K_2}$ by again using the procedure for the single Pitman-Yor process. A partition following the $\mathcal{ES}_{K_2}(d_1, c_1)$ distribution is produced via the CRP. This is represented in Figure 1 by the restaurant corresponding to $\mathcal{G}_1$. We denote the number of tables in the parent restaurant as K_1. Each of the K_1 tables is endowed with a parameter $\{\psi_{1k}\}_{k=1}^{K_1}$ drawn independently from $\mathcal{G}_0$. We set $\psi_{2k} = \psi_{1m}$ if in the parent restaurant customer k is sitting at table m and we set $\theta_j = \psi_{2k}$ if in the child restaurant customer j is sitting at table k. The recursive application of the CRP in a hierarchical model is known as the Chinese restaurant franchise (CRF) (Teh et al., 2006). The number of child restaurants is not restricted to one, as indicated by the dashed lines in Figure 1. Furthermore, the recursive nature of the CRF makes extensions to deeper hierarchies straightforward. For more detail refer to (Teh et al., 2006; Teh, 2006).

2.3. Sequence Memoizer

The sequence memoizer (SM) (Wood et al., 2009) is a model for discrete sequence data based on an unbounded depth hierarchical Pitman-Yor process. We can write the model as:

$$\begin{aligned} \mathcal{G}_{[]}|\mathcal{U}_\Sigma, d_0 &\sim \mathcal{PY}(d_0, 0, \mathcal{U}_\Sigma) \\ \mathcal{G}_{\mathbf{u}}|\mathcal{G}_{\sigma(\mathbf{u})}, d_{|\mathbf{u}|} &\sim \mathcal{PY}(d_{|\mathbf{u}|}, 0, \mathcal{G}_{\sigma(\mathbf{u})}) \quad \forall \mathbf{u} \in \Sigma^+ \\ \theta_n|\theta_{n-1}\dots\theta_1 = \mathbf{u} &\sim \mathcal{G}_{\mathbf{u}} \end{aligned}$$

where $\mathcal{U}_\Sigma$ is a uniform distribution over the set of symbols, $\mathbf{u}$ is a particular context, Σ^+ is the set of all such contexts, and $\sigma(\mathbf{u})$ is the context $\mathbf{u}$ modified by removing the most distant symbol. We assume $|\Sigma| < \infty$. Figure 2 shows the graphical model instantiated by the sequence *patat*. Note that in the SM graphical model, nodes that are not branching nodes and are not associated with observed data are not instantiated. The node labeled $\mathcal{G}_{patat}$ is only shown in the graphical model to indicate that the next symbol in the sequence will come from this distribution.

Inference in the SM model is performed in the CRF respresentation. Inference takes worst case $\mathcal{O}(n^2)$ time and requires $\mathcal{O}(n)$ space where n is the length of the sequence (Gasthaus et al., 2010). Quadratic time stems from the fact that seating a customer in the appropriate restaurant may require seating a customer in all of the restaurants above it. The length of this path is bounded by the length of the sequence. Each restaurant requires constant space because a restaurant need only maintain a constant number of summary statistics, the total number of customers and the total number of tables present of each type.[1] Note this representation requires reinstantiation of the full restaurant state for some steps of inference; this can be done by exploiting exchangeability. The fact that a restaurant can be represented in constant space allows the full model to be represented in $\mathcal{O}(n)$ space.

3. Theory

Towards developing a constant space HPYP model and inference procedure we first review dependent PY processes in Section 3.1. We then develop a dependent HPYP in Section 3.2. In both of these processes dependency arises from operations on customers in the CRP representation. We define our dependent HPYP in terms of operations on restaurants.

3.1. Dependent Pitman-Yor process

The $\mathcal{ES}_N(d, c)$ distribution discussed in Section 2 has an important consistency property. In the Chinese restaurant metaphor this consistency property corresponds to the fact that if a customer is removed uni-

[1] For symbol sets of finite cardinality.

formly at random, the remaining customer configuration represents a partition of the first $N-1$ integers following the $\mathcal{ES}_{N-1}(d,c)$ distribution (Pitman, 1995). This removal of a customer is known as a deletion operation. Another deletion operation, known as size-biased deletion, is one in which a customer is chosen uniformly at random and all customers seated at the same table are removed from the restaurant. Size biased deletion is known to satisfy a consistency property for the one parameter Ewen's distribution (Kingman, 1978), but the same is not true for the two parameter case (Pitman, 1995).

This consistency property is exploited in a generative procedure defined in the restaurant representation to draw samples $\{\{\theta_j^t\}_{j=1}^{N_t}\}_{t=1}^{T}$ from a sequence of dependent random distributions $\{\mathcal{G}^t\}_{t=1}^{T}$ such that

$$\begin{aligned} \mathcal{G}^t|d,c,\mathcal{G}_0 &\sim \mathcal{PY}(d,c,\mathcal{G}_0), \quad t = 1,\ldots,T \\ \theta_j^t|\mathcal{G}^t &\sim \mathcal{G}^t, \qquad\qquad\quad j = 1,\ldots,N_t. \end{aligned} \tag{1}$$

The variable t indexes the sequence. It may be useful to think of t as time though the sequence dependence need not be in time. The fact that each $\mathcal{G}^t$ in the sequence has the same distribution is known as stationarity (Brockwell & Davis, 1991).

A generative procedure that uses the first consistency property has been developed for dependent Dirichlet process models (Caron et al., 2007a). The generative procedure works for Pitman-Yor process models as well. It starts with an empty restaurant and generates $\{\theta_j^1\}_{j=1}^{N_1}$ using the CRP as usual. The process for generating $\{\theta_j^2\}_{j=1}^{N_2}$ is different than that used to generate $\{\theta_j^1\}_{j=1}^{N_1}$ in that the CRP does not start with an empty restaurant. The restaurant is instead initialized by retaining some of the customers from the restaurant representation for $\mathcal{G}^1$. Once new customers have been seated, previously unoccupied tables are endowed with a parameter ψ_k drawn independently from $\mathcal{G}_0$. We set $\theta_j^2 = \psi_k$ if the j^{th} customer seated during the second time step is seated at table k.

If l customers are deleted after time step 1 the partition represented by the starting customer configuration of the restaurant at time step 2 follows a $\mathcal{ES}_{N_1-l}(d,c)$ distribution. Therefore, after seating new customers in time step 2 the configuration results in a partition that follows a $\mathcal{ES}_{N_1-l+N_2}(d,c)$ distribution. Furthermore, the tables are endowed with parameter values drawn independently from $\mathcal{G}_0$. This fact confirms that the customer deletion process as described does generate samples from the model specified by Eqn. 1. Note that the number of customers deleted from the restaurant between time steps is independent of the consistency result and can thus be either stochastic or deterministic. For example one could choose to remove all customers in a restaurant.

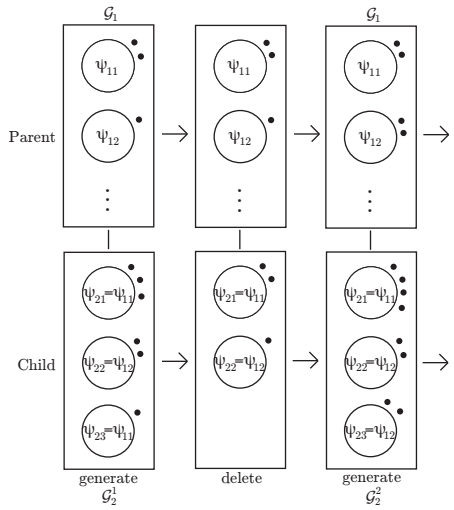

Figure 3. Depiction of dependent hierarchical Pitman-Yor process in Chinese restaurant franchise representation

If customers remain from one time step to the next then the G^t's are dependent. An exact characterization of the dependence is non-trivial. It has been shown in the analogous procedure for dependent Dirichlet processes that removing fewer customers between time steps induces higher dependence (Caron et al., 2007a). If no customers are deleted, $\mathcal{G}^t$ does not vary with the index t. If all customers are deleted between time steps t and $t+1$, $\mathcal{G}^t$ and $\mathcal{G}^{t+1}$ are independent conditional on $\mathcal{G}_0$. In designing a dependent HPYP we will exploit this last characteristic.

3.2. Dependent HPYP

The mechanism for generating dependent PY processes can be used to generate dependent HPYP's as well. We develop one such strategy for doing so here. Consider the following two level model:

$$\begin{aligned} \mathcal{G}_1|d_1,c_1,\mathcal{G}_0 &\sim \mathcal{PY}(d_1,c_1,\mathcal{G}_0) \\ \mathcal{G}_2^t|d_2,c_2,\mathcal{G}_1 &\sim \mathcal{PY}(d_2,c_1,\mathcal{G}_1), \quad t = 1,\ldots,T \\ \theta_i^t|\mathcal{G}_2^t &\sim \mathcal{G}_2^t, \qquad\qquad\quad i = 1,\ldots,N_t. \end{aligned} \tag{2}$$

A procedure that can be used to generate data from this model can be obtained by combining the Chinese restaurant franchise with the customer deletion scheme from Section 3.1. Note that the model specification indicates that $\mathcal{G}_1$ does not vary with the index t. This means that deletion of customers only occurs in the child restaurant.

Figure 3 shows a sample from the model given in Eqn. 2 generated using the procedure described in this section. The middle column of restaurants show the restaurant configurations after a deletion step. Note the configuration of the parent restaurant does not change even though one of the tables in the child restaurant has been removed. The restaurant configurations in the third column show the resulting seating arrangement after the sample $\{\theta_j^2\}_{j=1}^5$ was generated. Here, a customer was added to the the parent restaurant in order to generate the parameter ψ_{12} given to the third table in the child restaurant.

Extending the generative procedure to draw samples from dependent HPYP's of the type

$$\begin{aligned} \mathcal{G}_1^t|d_1,c_1,\mathcal{G}_0 &\sim \mathcal{PY}(d_1,d_2,\mathcal{G}_0) \\ \mathcal{G}_2^t|d_2,c_2,\mathcal{G}_1^t &\sim \mathcal{PY}(d_2,c_2,\mathcal{G}_1^t), \quad t=1,\ldots,T \\ \theta_i^t|\mathcal{G}_2^t &\sim \mathcal{G}_2^t, \quad i=1,\ldots,N_t \end{aligned} \tag{3}$$

can be done. Dependence is induced between the $\mathcal{G}_1^t$'s by using the deletion scheme in the parent restaurants. In order to produce a sample from the model described by Eqn. 3 the configuration of the child restaurants must also be altered. If we assume independence of the $\mathcal{G}_2^t$'s then the appropriate action is to delete all customers in the child restaurant between time steps. This is the action we are going to take. Without the assumption of independence the extension is not straightforward. It is likely that such a process exists, but we do not develop it in this paper.

While the theory developed from this point forward is applicable to all HPYP models we restrict our attention to the SM model. This is because we are specifically interested in using the deletion process to control the memory complexity of inference in the SM.

3.3. Constant memory

The deletion mechanism used to define the dependent HPYP can alternatively be viewed as a way to limit the space complexity required to estimate the SM. As noted earlier, the number of instantiated restaurants is the limiting factor regarding memory usage. For this discussion we will therefore consider a single instantiated restaurant as a unit of memory. For the deletion scheme to limit the amount of memory used in the SM, we must be able to limit the number of instantiated restaurants.

The number of instantiated restaurants can be limited using the deletion scheme developed for dependent HPYP's. Memory savings can be achieved by deleting all the customers in a leaf restaurant since leaf restaurants without people do not need to be represented. We call a restaurant a leaf restaurant if all restaurants descended from it are empty. Nodes corresponding to leaf restaurants are shaded in Fig. 2.

The theory developed in Section 3.1 aids in understanding the implied dependencies that arise from deleting customers in leaf restaurants. For example, in Fig. 2 consider what would happen if we deleted all the customers in the restaurant labeled $\mathcal{G}_{pa}$. Let $\mathcal{G}_{pa}^t = \mathcal{G}_{pa}$. Implicit in this deletion is the assumption that the distribution over symbols following the context pa in the sequence prior to the deletion step, G_{pa}^t is, conditioned on $\mathcal{G}_a$, independent of the distribution $\mathcal{G}_{pa}^{t+1}$ after the deletion at time t.

Returning to the SM model and the deletion of leaf resaurants, we note that the parent restaurant of a leaf restaurant may not be present in the SM tree. An example is the restaurant $\mathcal{G}_{pata}$ in Figure 2. The parent restaurant of $\mathcal{G}_{pata}$ is $\mathcal{G}_{\sigma(pata)} = \mathcal{G}_{ata}$. The parent in the SM tree is $\mathcal{G}_{\pi(pata)} = \mathcal{G}_a$.

To attain memory savings by deleting restaurant $\mathcal{G}_{pata}$ we must also delete all of the restaurants in the path from $\mathcal{G}_a$ to $\mathcal{G}_{pata}$. While the restaurants on this path are not instantiated in the representation of the model, they do contain customers and their deletion is significant. The implicit model assumption made by deleting $\mathcal{G}_{pata}$ is that the distributions in the subtree including and below $\mathcal{G}_{ta}$ before the deletion step and the same distributions after the deletion step are independent.

This is the basic framework for both bounded memory algorithms we present. That the model can vary over time may be appropriate for very long sequences. Our deletion process requires us to assume the independence of a large number of distributions. While this may seem troubling, a key insight is that the entire set of distributions for which we must make the independence assumption are dependent on an existing node in the tree. Much of the information will be retained in the tree by the remaining node.

Finally, we point out that the theory behind these deletion operations holds for general hierarchical Pitman-Yor processes and thus also for finite depth n-gram style models. In Section 5 we show some results concerning this type of model as well.

4. Inference

In Section 3 we defined a generative model for a dependent HPYP. In this section we consider inference.

The sequential nature of the generative process suggests using sequential Monte Carlo (SMC) for inference (Doucet et al., 2001). In Algorithm 1 we show

Algorithm 1 Particle Filter

Initialize K particles $\{\{R^k = \emptyset, w^k = \frac{1}{K}\}\}_{k=1}^K$
Initialize $\mathbf{u} = []$
while true **do**
  $x \leftarrow$ NextObservation()
  $\mathbf{u} \leftarrow \mathbf{u}x$
  for all $k = 1, \ldots, K$ **do**
    if $|R^k| >$ maxRestaurants $- 2$ **then**
      $R^k_{\mathbf{d}} \leftarrow$ ProposeRestaurantToDelete()
      $R^k \leftarrow R^k \setminus R^k_{\mathbf{d}}$
    end if
    $\{R^k_{\mathbf{u}}, R^k_{\pi(\mathbf{u})}, \ldots, R^k_{[]}\} \leftarrow$ FindRestaurant($\mathbf{u}$)
    $\mathbf{P}^k \leftarrow \{R^k_{\mathbf{u}}, R^k_{\pi(\mathbf{u})}, \ldots, R^k_{[]}\}$
    $R^k \leftarrow R^k \cup \mathbf{P}^k$
    $v^k \leftarrow$ Seat($x, \mathbf{P}^k$)
    $w^k \leftarrow v^k w^k$
  end for
  # resample particles
  # predict
end while

a SMC procedure for estimating the SM reviewed in Section 2.3 in which the restaurant emptying dependency mechanism of Section 3.2 is employed. In this algorithm each particle consists of a set of occupied restaurants and a weight. Each particle is updated after each symbol in a streaming sequence.

NextObservation() returns the next symbol in the sequence. The variable $\mathbf{u}$ is the context in which the symbol x was observed. ProposeRestaurantToDelete() returns a restaurant to empty from proposal distribution to be defined. FindRestaurant($\mathbf{u}$) returns a path from the restaurant $\mathcal{G}_{[]}$ to the restaurant $\mathcal{G}_{\mathbf{u}}$. Finding the path requires descending the tree from the root and instantiating all necessary restaurants. The set of instantiated restaurants in the particle is updated to include restaurants in the FindRestaurant($\mathbf{u}$) path. Seat($x, \mathbf{P^k}$) returns the a priori probability of observing x in the context $\mathbf{u}$ and seats a customer in the restaurant $\mathcal{G}_{\mathbf{u}}$ according to the posterior distribution over tables given the state of the particle and the observation x. If Seat() sits the customer associated with x at a new table it will also seat a customer in the parent restaurant. This seating process may extend all the way up to the restaurant $\mathcal{G}_{[]}$.

Most steps of the particle filter are based on the estimation approach for the SM proposed in (Gasthaus et al., 2010). However, this inferential procedure must include the deletion step characteristic of the generative model. When the state of the model represented in each particle reaches the memory constraint (maxRestaurants) we need to propose which restaurants to delete. We suggest two different proposal distributions to correspond with the two interpretations of the model specification already discussed. The first corresponds to the interpretation of the model as a sequence of dependent distributions with Pitman-Yor process distributions which vary sequentially across the sequence. The second corresponds to the interpretation of the model as a finite state approximation of a model representation which grows linearly as a function of the length of the data.

The first proposal distribution for the deletion of restaurants is uniform over the leaf nodes of the current state of the model. Using the generative process as the proposal distribution is standard in particle filtering approaches (Doucet et al., 2001). We refer to the use of this proposal distribution as "random deletion" in Figure 4 and elsewhere.

For the second proposal distribution we note that the fixed state of the model can be used to approximate the probability of the whole sequence. Furthermore, by deleting different leaf restaurants the probability of the sequence, given the updated state of the model, changes. The second proposal distribution deterministically proposes the leaf restaurant whose deletion least negatively impacts the estimated likelihood of the observed sequence. We refer to the use of this proposal distribution as "greedy deletion".

5. Experiments and Results

In tests of predictive probability using incremental inference, the constant space deletion schemes outperformed commercial compressors and simpler constant space models, while approaching the performance of linear space methods even with significantly less space. The data we used in our experiments was the Calgary Corpus (CC) (Bell et al., 1989). The CC is a compression benchmark corpus that consists of 14 files chosen to cover a typical range of file types. A separate instance was trained for every combination of model tested (naive, finite depth, random deletion, greedy deletion) and file. Files were treated as a sequences of bytes. Estimation was performed incrementally using Algorithm 1 with $K = 1$. The posterior predictive probability of each byte was calculated before that byte was incorporated into the model. The random number generator was initialized with a new seed for every instance at the start of each run. Figure 4 shows the average predictive performance of several sequential prediction models (error bars were too small to be included). The reported results are the per-byte log-loss in bits averaged over all files in the corpus.

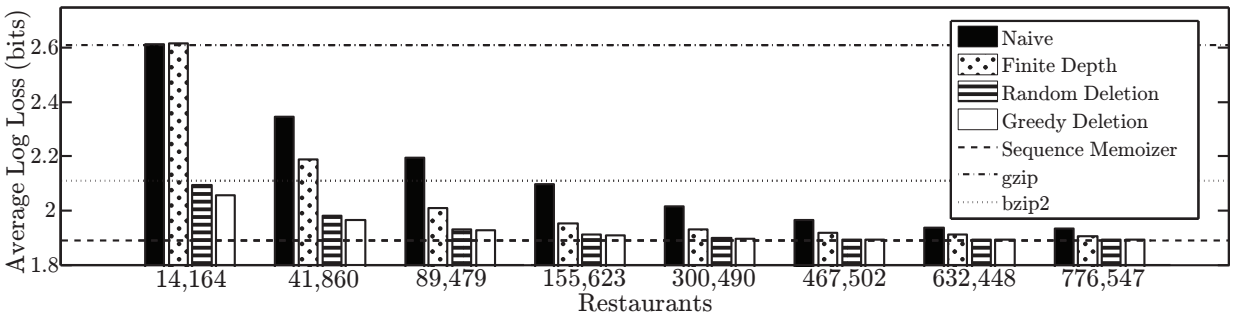

Figure 4. Performance comparison of constant memory sequence models of the Calgary Corpus

To evaluate our two procedures for constant space inference, we compared to the full SM, two other simple constant space versions of the SM and two standard compressors. The SM does not delete any restaurants and incremental inference is linear in space and time. The two inference procedures of interest (random and greedy deletion) are both schemes for deleting restaurants from the model, one which randomly deletes restaurants and one which greedily deletes the restaurants which make the smallest change possible to the approximate likelihood of the data already observed, as described in Section 4. Inference in the naïve constant space SM runs the same as for a normal SM until the maximum number of restaurants is reached, at which point all restaurants are removed and inference continues along the remainder of the sequence. The finite depth model takes the SM and bounds the length of a preceding sequence included in an observation. Thus, it is a finite depth SM. Since we cannot know the number of restaurants in a finite depth model before estimation, we perform inference with this model first and use the number of restaurants to limit the space allowed by other models. This means that the deletion strategies must actually choose a better graphical model in order to outperform the finite depth model with the same number of nodes. We also compressed each file using gzip and bzip2 (Deutsch et al., 1996; Seward, 1999). Default parameters were used for the commercial compressors.

We first modeled each file with a finite depth SM model and recorded the number of instantiated restaurants. The model complexity was allowed to grow without bound, meaning the finite depth model was at its full representational capacity. The maximum number of instantiated restaurants gave an upper bound on the number of restaurants needed to model any file in the corpus with an HPYP model of a given depth. We calculated this upper bound for depths 2 through 9. This set of bounds was then used as the maximum number of restaurants for the naive model and the random and greedy deletion models. Results are shown in Figure 4. For comparison, the full SM model instantiated 1,160,765 restaurants when trained on the same documents.

Each of the four sequence models were fit ten times. From that the variance of the inference procedure was estimated. Our results show the standard deviation of the average log loss to be less than 0.002 for all of the methods. All differences detectable in the graph are significant at the $\alpha = 0.01$ level. We note that the SM model paired with either deletion scheme consistently outperforms the finite depth model and the naïve model. We also note that both deletion schemes approach the performance of the full SM with only one sixth as many restaurants. Furthermore, the greedy deletion scheme shows improvement over the random deletion scheme, especially for tight memory constraints. The SM model paired with either deletion scheme outperformed the commercial compressors at every threshold tested.

6. Discussion

In the course of this paper we developed a generative model for a dependent HPYP. This model is amenable to sequential Monte Carlo estimation. We know that other ways to generate dependent HPYP's will emerge, and hope that others too will be amenable to efficient incremental inference techniques. We explained how dependency in our specific dependent HPYP formulation arises from deleting whole restaurants in a Chinese restaurant franchise representation. We show that this restaurant deletion scheme can be either be deterministic or probabilistic. In contrast to others who have used deletion in Chinese restaurant representations to induce dependence between processes we suggested using a deterministic deletion policy to control the memory complexity of inference. The utility of our predictive model seems promising, particularly when considering how it might be used in a probabilistic lossless

compressor. This is because the performance of the constant memory models is nearly indistinguishable from that of state of the the art sequence models and significantly better than that of existing commercial compressors.

The techniques highlighted in this paper point to interesting avenues for future research. In essence the restaurant forgetting scheme amounts to a greedy stochastic approach to graphical model structure learning. As data continually arrives, only those graphical model nodes corresponding to contexts that are both meaningful and continually reappear will remain in the graphical model after many deletion operations. While the deletion scheme utilized in this work is highlighted in the context of a single family of models corresponding to a single graphical model architecture, it may be possible to use deletion operations in models of more general architecture.

References

Bell, T., Witten, I.H., and Cleary, J.G. Modeling for text compression. *ACM Computing Surveys (CSUR)*, 21(4):557–591, 1989.

Blackwell, D. and MacQueen, J. Ferguson distributions via Polya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

Brockwell, P.J. and Davis, R.A. *Time series: theory and methods*. Springer, 1991.

Caron, F., Davy, M., and Doucet, A. Generalized polya urn for time-varying Dirichlet process mixtures. In *23rd Conference on Uncertainty in Artificial Intelligence (UAI'2007), Vancouver, Canada, July 2007*, 2007a.

Caron, F., M., Davy, and Doucet, A. Stationary Pitman-Yor processes. Poster, Isaac Newton Institute Workshop on Construction and Properties of Bayesian nonparametric regression models, August 2007b.

Csató, L. and Opper., M. Sparse online Gaussian processes. *Neural Computation*, pp. 641–668, 2002.

Deutsch, P. et al. GZIP file format specification version 4.3, 1996.

Doucet, A., de Freitas, N., and Gordon, N. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

Ewens, WJ and Tavare, S. The Ewens sampling formula. *Multivariate Discrete Distributions. Wiley, New York*, 1995.

Gasthaus, J., Wood, F., and Teh, Y. W. Lossless compression based on the Sequence Memoizer. In *Data Compression Conference 2010*, pp. 337–345, 2010.

Griffin, J.E. and Steel, M.F.J. Order-based dependent Dirichlet processes. *Journal of the American Association of Statistics*, 101(473):179–194, 2006.

Kingman, JFC. Random partitions in population genetics. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 361 (1704):1–20, 1978.

Lawrence, N., Seeger, M., and Herbrich, R. Fast sparse Gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pp. 625–632, 2003.

MacEachern, S. N. Dependent Dirichlet processes. Technical report, Department of Statistics, Ohio State University, 2000.

Pitman, J. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102(2):145–158, 1995.

Pitman, J. and Yor, M. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900, 1997.

Seward, J. bzip2. *Go online to* `http://www.spec.org/osg/cpu2000/CINT2000/-256.bzip2/docs/256.bzip2.html`, 1999.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pp. 1257–1264. MIT press, 2006.

Srebro, N. and Roweis, S. Time-varying topic models using dependent Dirichlet processes. Technical Report UTML-TR-2005-003, Department of Computer Science, University of Toronto, 2005.

Teh, Y. W. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the Association for Computational Linguistics*, pp. 985–992, 2006.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476): 1566–1581, 2006.

Wood, F., Archambeau, C., Gasthaus, J., James, L., and Teh, Y. W. A stochastic memoizer for sequence data. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 1129–1136, Montreal, Canada, 2009.