Filtering with Abstract Particles

Jacob Steinhardt JSTEINHARDT@CS.STANFORD.EDU
Percy Liang PLIANG@CS.STANFORD.EDU
Stanford University, 353 Serra Street, Stanford, CA 94305 USA

Abstract

By using particles, beam search and sequential Monte Carlo can approximate distributions in an extremely flexible manner. However, they can suffer from sparsity and inadequate coverage on large state spaces. We present a new filtering method for discrete spaces that addresses this issue by using "abstract particles," each of which represents an entire region of state space. These abstract particles are combined into a hierarchical decomposition, yielding a compact and flexible representation. Empirically, our method outperforms beam search and sequential Monte Carlo on both a text reconstruction task and a multiple object tracking task.

1. Introduction

Sequential Monte Carlo (Cappé et al., 2007; Doucet & Johansen, 2011), together with its deterministic analogue beam search (Pal et al., 2006) have been incredibly successful at solving a wide variety of filtering and other inference tasks (Quirk & Moore, 2007; Görür & Teh, 2008; Carreras & Collins, 2009; Liang et al., 2011; Bouchard-Côté et al., 2012). However, despite their flexibility, they can perform poorly when there are insufficiently many particles to cover all high-probability regions of the posterior distribution; it is then possible that *all* particles will assign low probability to a new observation, which leads to estimates with very high variance (Gilks & Berzuini, 2001).

The issue is a lack of *coverage*: when there are relevant regions of the space that are not represented at all by the particles at hand. Motivated by this observation, we construct "abstract particles" where each particle covers an entire region of the space. Within each particle we then perform a *local* variational approximation to the target distribution over that region (Jordan et al., 1999; Minka, 2001). More importantly, we also optimize over the *choice* of regions. This can be thought of as adaptively choosing the *structure* of the variational family. Our contributions are four-fold; in all cases we are focused on inference in discrete spaces.

- We formally define the notion of an abstract particle (Section 2.1).
- We identify a large but tractable family of abstract particles based on hierarchical decompositions (Section 2.2).
- We provide efficient algorithms for choosing particles from this family (Section 4).
- We demonstrate improved performance over both beam search and sequential Monte Carlo on two tasks (Section 5).

2. Relevance-Directed Variational Approximation

Our goal is to approximate an intractable target distribution $p^*(x)$ by a tractable distribution $\hat{p}(x)$. We assume that p^* is given as an unnormalized product of factors. A common approach would be to parameterize $\hat{p}$ as an exponential family:

$$\hat{p}_\theta(x) \stackrel{\text{def}}{=} \frac{1}{Z(\theta)} \exp(\theta^\top \phi(x)) \tag{1}$$

and then choose θ to minimize the objective function

$$\mathrm{KL}\left(p^* \parallel \hat{p}_\theta\right) \stackrel{\text{def}}{=} \sum_x p^*(x) \log \left(\frac{p^*(x)}{\hat{p}_\theta(x)}\right). \tag{2}$$

In practice, finding the optimum of (2) is intractable, but one can still employ heuristic techniques such as expectation propagation (Minka, 2001).

Such variational approximations are useful in allowing us to approximate otherwise intractable distributions; however, they suffer in that they are forced to approximate the entire space by a single, coarse distribution. Sometimes, some regions of the space are more important than others, and we would like a way to capture this in our approximation.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

To do so, we will show how to obtain improved variational approximations by partitioning the state space, then provide a formalism for optimizing the partition via hierarchical decompositions.

2.1. Improved Variational Approximations via Partitions

Let $Y = \{Y_1, \ldots, Y_k\}$ be any partition of $\mathcal{X}$. Given an exponential family $\mathcal{F}$ defined by sufficient statistics ϕ, define $\mathcal{F}^Y$ to be the exponential family with sufficient statistics $\{\phi \mathbb{I}_{Y_j}\}_{j=1}^k$, which corresponds to taking the outer product of ϕ with the indicator functions of the partition. $\mathcal{F}^Y$ is thus an expanded variational family that is more flexible than $\mathcal{F}$ itself. Using $\theta = (\theta_j)_{j=1}^k$ to indicate the parameters of $\mathcal{F}^Y$, the resulting distributions are piecewise exponential families on each element of the partition, with a single global normalization constant $Z(\theta)$:

$$\hat{p}_\theta(x) = \frac{1}{Z(\theta)} \exp(\theta_j^\top \phi(x)) \ : \ x \in Y_j \tag{3}$$

(this is similar to the setup of split variational inference in (Bouchard & Zoeter, 2009)). We refer to each Y_j as an *abstract particle*. Define $\hat{f}_{\theta_j}$ to be the unnormalized probability $\exp(\theta_j^\top \phi(x))$ over Y_j. Because Y is a partition of $\mathcal{X}$, the KL divergence decouples over each $Y_j \in Y$:

$$\mathrm{KL}\left(p^* \parallel \hat{p}_\theta\right) = \sum_{j=1}^k \sum_{x \in Y_j} p^*(x) \log \left(\frac{p^*(x)}{\frac{1}{Z(\theta)} \hat{f}_{\theta_j}(x)} \right) \tag{4}$$

$$= \log Z(\theta) + \sum_{j=1}^k \mathrm{KL}_{Y_j}\left(p^* \,\middle\|\, \hat{f}_{\theta_j}\right). \tag{5}$$

Here $\mathrm{KL}_S\left(p \parallel q\right) \stackrel{\text{def}}{=} \sum_{x \in S} p(x) \log\left(\frac{p(x)}{q(x)}\right)$ is the *region-specific KL divergence*. Note that minimizing KL in this direction is intractable; this is a general property of the problem, not a consequence of introducing a partition.

There are many techniques for heuristically minimizing the KL, such as expectation propagation (Minka, 2001). Another interesting approach (though it has no justification in terms of variational inference) is to obtain the $\hat{f}_{\theta_j}$ by using a fixed simpler model which has been trained on the same data as p^* (we will explain this in more detail later). Normally, using only a subset of features of the original model p^* would result in a poor approximation. However, when combined with partitions, this approach can be very powerful, as we will demonstrate in a text reconstruction experiment (see Section 3.1).

For now, we will simply assume that we have a function `Fit` that maps Y_j to θ_j. We will explore both the use of EP and of lower-order models in Section 3.

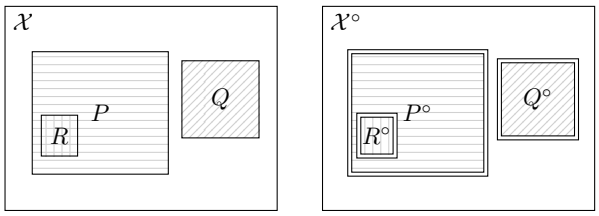

Figure 1. A hierarchical decomposition (left) and its corresponding partition (right). We have $\mathcal{C}(\mathcal{X}) = \{P, Q\}$, $\mathcal{C}(P) = \{R\}$, and $\mathcal{C}(Q) = \mathcal{C}(R) = \emptyset$. The elements of the partition are $\mathcal{X}^\circ = \mathcal{X} \backslash (P \cup Q)$, $P^\circ = P \backslash R$, $Q^\circ = Q$, and $R^\circ = R$.

2.2. Optimizing the Partition via Hierarchical Decompositions

While a fixed partition Y allows us to construct a richer exponential family $\mathcal{F}^Y$ out of $\mathcal{F}$, a more interesting prospect is to adaptively optimize the partition Y. The tool we will use for doing so is a *hierarchical decomposition*:

Definition 2.1. Given a space $\mathcal{X}$, a hierarchical decomposition is a collection $A \subseteq 2^{\mathcal{X}}$ of subsets satisfying: (i) $\mathcal{X} \in A$ and (ii) if $a, b \in A$, then $a \cap b \in \{a, b, \emptyset\}$.

Note that the elements of A form a tree. For $a \in A$, let $\mathcal{C}_A(a)$ denote the children of a in the tree: $b \in \mathcal{C}_A(a)$ iff $b \subsetneq a$ and there is no $b' \in A$ with $b \subsetneq b' \subsetneq a$. An example of a decomposition is shown in the left panel of Figure 1.

Finally, let a° for $a \in A$ denote the set of elements lying in a but not in any of its children:

$$a^\circ \stackrel{\text{def}}{=} a \backslash \left[\bigcup_{b \in \mathcal{C}_A(a)} b \right]. \tag{6}$$

This is illustrated in the right panel of Figure 1. Hierarchical decompositions are connected to partitions in the following way:

Lemma 2.2. *If A is a hierarchical decomposition, then the collection $A^\circ \stackrel{\text{def}}{=} \{a^\circ\}_{a \in A}$ forms a partition of $\mathcal{X}$.*

Thus, every hierarchical decomposition gives rise to a partition, as does any subset:

Lemma 2.3. *Let A be a hierarchical decomposition and let $B \subseteq A$ such that $\mathcal{X} \in B$. Then B is a hierarchical decomposition.*

Therefore, a hierarchical decomposition A determines a family of $2^{|A|-1}$ partitions (some of which may end up being identical), defined by

$$\{B^\circ \mid \{\mathcal{X}\} \subseteq B \subseteq A\}. \tag{7}$$

We can now cast the optimization of the partition as the following problem: *Given a (large) hierarchical decomposition A, find a subset B of (small) size k and a distribution $\hat{p} \in \mathcal{F}^{B^\circ}$ such that $\mathrm{KL}\left(p^* \parallel \hat{p}\right)$ is small.*

In equations, this is

$$\begin{aligned} &\text{minimize } \mathrm{KL}\left(p^* \parallel \hat{p}\right) \\ &\text{subject to } \hat{p} \in \mathcal{F}^{B^\circ} \\ &\qquad \{\mathcal{X}\} \subseteq B \subseteq A \\ &\qquad |B| \leq k \end{aligned} \tag{8}$$

Note that $\hat{p}$ will be parameterized by a variable $\theta = (\theta_a)_{a \in B}$. While there are exponentially many possibilities for B, we can heuristically optimize (8) over B using a greedy algorithm, as well as exactly optimize a surrogate function using dynamic programming. The greedy algorithm is described in Section 4 and the dynamic programming algorithm is described in the supplementary material. In Section 4 we also provide an algorithm, analogous to beam search, for approximately optimizing B in cases where A itself is exponentially large.

Effect of hierarchical decomposition on approximation. As an example to better understand how the hierarchical decomposition affects the approximation, suppose that $\mathcal{F}$ consists of distributions over (x_1, x_2) that do not depend on x_1: $\hat{p}(x_1, x_2) = q(x_2)$. Let a be a region where x_1 is fixed to a value $\hat{x}_1$. Then we can take $\hat{f}_{\theta_a}(x_1, x_2) = p^*(\hat{x}_1)p^*(x_2 \mid \hat{x}_1)$. Since $\hat{x}_1$ is a constant, $\hat{f}_{\theta_a}$ doesn't "depend" on x_1, even though it matches p^* exactly over a.

More generally, any factor of p^* that is constant over a region a can be matched exactly over that region, no matter what the family $\mathcal{F}$. This provides some intuition for the role that the hierarchical decomposition B plays in improving the approximation to p^*: even if $\mathcal{F}$ cannot approximate a factor satisfactorily, an appropriately chosen hierarchical decomposition will be able to incorporate the factor exactly, at least locally.

Note that in the limiting case where a region a consists of a single point, all factors are incorporated exactly, and we recover regular particle-based inference.

Computational issues. One important question is which hierarchical decompositions A lead to a tractable family $\mathcal{F}^{B^\circ}$. Typically we will want to choose A so that each region a is "simple", in order to compute the normalization constant $Z(\theta)$. To elaborate, let $\hat{f}_{\theta_a}$ denote the *unnormalized* exponential family $\exp(\theta_a^\top \phi(x))$, and note the equalities

$$Z(\theta) = \sum_{a \in B} \sum_{x \in a^\circ} \hat{f}_{\theta_a}(x) \tag{9}$$

$$= \sum_{a \in B} \left[\hat{f}_{\theta_a}(a) - \sum_{b \in \mathcal{C}_B(a)} \hat{f}_{\theta_a}(b) \right], \tag{10}$$

which allows us to compute $Z(\theta)$ as long as we can compute $\hat{f}_{\theta_a}(b)$ for given regions $b \in A$. If b is defined by fixing the values of certain variables in the model, then $\hat{f}_{\theta_a}(b)$ is just an (unnormalized) marginal probability and so is often tractable. Note that the KL divergence decomposes similarly to (10), although the individual terms in the decomposition will not be tractably computable in general.

In practice, we will never work directly with a° from a computational perspective, but rather use a and its children $\mathcal{C}_B(a)$ together with equation (10). To make this clearer notationally, we will use $\texttt{Fit}(a, \mathcal{C}_B(a))$ to denote the value obtained for θ_a (in fact, in all of our later examples θ_a depends only on a).

2.3. Summary

We have so far presented a recipe for improving variational approximations via partitions of the space, using hierarchical decompositions to provide an interesting family of partitions. The recipe consists of the following steps:

1. Take as input a target distribution p^*.
2. Choose a variational family $\mathcal{F}$ and a hierarchical decomposition A.
3. Choose a fitting method $\texttt{Fit}(a, \mathcal{C}(a))$.
4. Find a subset B of A of size k by optimizing (8).

This leaves us with two outstanding issues, which we address in subsequent sections:

- How to pick $\mathcal{F}$, A, and $\texttt{Fit}$ for two concrete filtering problems (Section 3).
- How to find a good choice of B (Section 4), since in practice (8) can only be solved approximately.

3. Examples

We have seen how a variational family $\mathcal{F}$, together with a hierarchical decomposition of the space $\mathcal{X}$, can in theory be used to construct a much richer variational family. In this section, we consider two filtering tasks, showing how hierarchical decompositions are more powerful than standard particle-based methods. In each case, we will define a generative model $p(x, y) = p(x)p(y \mid x)$, and consider the posterior inference problem $p^*(x) = p(x \mid y)$. Since y is observed, we will form our hierarchical decomposition over $\mathcal{X}$ only, rather than $\mathcal{X} \times \mathcal{Y}$.

We assume that $\mathcal{X}$ consists of ordered T-tuples $(x_1, \ldots, x_T) \in \Sigma^T$, where Σ is the alphabet. Our task is to predict x_t given $y_{1:t}$ for all $t = 1, \ldots, T$. For clarity of exposition, we focus on the $t = T$ case.

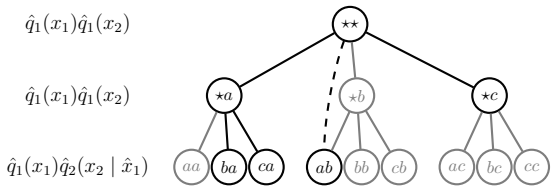

Figure 2. A hierarchical decomposition A (all nodes) together with a subset B (black). The underlying space $\mathcal{X}$ is $\{a, b, c\}^2$. The symbol $\star$ is a wildcard, so for instance $\star a$ indicates the set $\{aa, ba, ca\}$. Note that the parent of ab in B is $\star\star$, even though its parent in A is $\star b$. The local approximation to p^* used at each level of the decomposition is indicated on the left (see Equation 14).

3.1. Text Reconstruction

Setup. We consider the task of text reconstruction, where $x_{1:T}$ is a string of characters and each observation y_t is either present (in which case $y_t = x_t$) or missing (in which case $y_t =$ `'?'`). Our prior over $x_{1:T}$ is a character-level n-gram model, which has been used in named-entity recognition (Klein et al., 2003), authorship attribution (Kešelj et al., 2003), and language identification (Vatanen et al., 2010). Our generative model is as follows:

$$p(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{(t-n+1):(t-1)}) p(y_t \mid x_t). \tag{11}$$

Optimal performance for such models is typically achieved at relatively large values of n, around 5 or 6 (Vatanen et al., 2010; Klein et al., 2003). In such cases, exact inference can be quite expensive.

Approximating family ($\mathcal{F}$). Our variational family $\mathcal{F}$ consists of all unigram models. By itself, $\mathcal{F}$ would provide a very poor approximation to p^*. However, using a hierarchical decomposition A will provide a much better approximation.

Hierarchical decomposition (A). The 0th layer of the decomposition consists of a single set containing all of $\mathcal{X}$; the 1st layer partitions the space based on x_T; the 2nd layer partitions the space based on x_{T-1} and x_T; and so on. Recalling that Σ is the alphabet, the dth layer of the hierarchical decomposition A can be expressed algebraically as

$$\left\{\Sigma^{T-d} \times \{\hat{x}_{(T-d+1):T}\} \;\middle|\; \hat{x}_{(T-d+1):T} \in \Sigma^d\right\} \tag{12}$$

(so we abstract away $x_{1:T-d}$ but represent $x_{(T-d+1):T}$ concretely). A simple example is given in Figure 2.

Region-specific model (`Fit`). Suppose the region a is defined by a sequence $\hat{x}_{t_0:T}$ as in (12), where we use t_0 in place of $T - d + 1$ for convenience in the sequel.

To approximate $p^*(x) = p(x \mid y)$, we define the following unigram model, where $\hat{f}_{\theta_a}$ denotes the unnormalized probability over a°:

$$\hat{f}_{\theta_a}(x) = \prod_{t=1}^{T} \hat{\psi}_t(x_t). \tag{13}$$

Then $\hat{p}_\theta$ is *locally* a unigram model (equal to $\frac{1}{Z(\theta)} \hat{f}_{\theta_a}$) over each region a°; note however that the full distribution $\hat{p}_\theta$ has much richer structure. In particular, since x_{t_0} is fixed across a, we can let $\hat{\psi}_{t_0+1}(x_{t_0+1})$ "depend" on x_{t_0} by setting it equal to $\hat{q}_2(x_{t_0+1} \mid \hat{x}_{t_0})$, where $\hat{q}_2$ is a bigram model. Similarly, we can set $\hat{\psi}_{t_0+2}$ to $\hat{q}_3(x_{t_0+2} \mid \hat{x}_{t_0+1}, \hat{x}_{t_0})$, where $\hat{q}_3$ is a trigram model. In general, we will fit lower-order m-gram models $\hat{q}_m$ for $1 \le m < n$, and use the approximation

$$\hat{\psi}_t(x_t) = \begin{cases} \hat{q}_1(x_t) & : \; t \le t_0 \\ \hat{q}_{t-t_0+1}(x_t \mid \hat{x}_{t_0:(t-1)}) & : \; t_0 < t < t_0 + n - 1 \\ p(x_t \mid \hat{x}_{(t-n+1):(t-1)}) & : \; t \ge t_0 + n - 1 \end{cases} \tag{14}$$

(This is all for the case when y_t equals `'?'`; otherwise we would have $\hat{\psi}_t(x_t) = \delta(x_t = y_t)$.)

So, by picking an appropriate subset B of A, we can interpolate as necessary between the coarse unigram model and exact inference; for instance, if we take the entire Tth layer (i.e. all the singleton sets) then we recover the exact posterior. Note that this is *not* necessarily true if we only take the nth layer, despite the model being an n-gram model. This is because $\hat{\psi}_{T-n+1}(x_{T-n+1})$ would still be $\hat{q}_1(x_{T-n+1})$, rather than $p(x_{T-n+1} \mid \hat{x}_{(T-2n+2):(T-n)})$.

More generally, for any element of the dth layer, information is propagated exactly for the final d time steps, while the characters are treated completely independently (no propagation of information) for the first $T - d$ time steps. By optimizing the elements of the hierarchical decomposition, we identify regions where the information to propagate is important, and expend additional computational resources on those regions.

3.2. Multiple Object Tracking

Setup. Another filtering problem where exact inference is intractable is multiple object tracking. Suppose that we have K objects following trajectories through some space, and a sensor that can detect the location of an object but not which object it is; the object identities must therefore be disambiguated using knowledge about their dynamics. For simplicity, we assume that only one object is observed at a time; this isn't necessary for the math to work out.

We can model this as a factorial HMM; there are K independently evolving Markov chains

$x_{1,1:T}, x_{2,1:T}, \ldots, x_{K,1:T}$, together with independent hidden variables $c_1, \ldots, c_T \sim \text{Uniform}(\{1, \ldots, K\})$ such that c_t specifies which of the objects was observed at time t. The observed node, y_t, is deterministically equal to $x_{c_t,t}$. Assuming $x_{k,t}$ takes values in $[S] \stackrel{\text{def}}{=} \{1, \ldots, S\}$, the alphabet Σ is $[S]^K \times [K]$; thus $|\Sigma|$ grows exponentially in K.

Approximating family ($\mathcal{F}$). We will use independent Markov chains over each $x_{k,1:T}$ as our variational family $\mathcal{F}$. This avoids the computational difficulties of the true posterior, in which the observation potential $p(y_t \mid x_{1:K,t}, c_t)$ couples all of the chains.

Hierarchical decomposition (A). Our hierarchical decomposition A consists of regions where some suffix of the c_t are specified; that is, the dth level is

$$\left\{ \left([K]^{T-d} \times \{\hat{c}_{t_0:T}\} \right) \times [S]^{TK} \mid \hat{c}_{t_0:T} \in [K]^d \right\}, \quad (15)$$

with $t_0 = T - d + 1$.

Region-specific model (`Fit`). Suppose that our region a is defined by a sequence $\hat{c}_{t_0:T}$ as in (15). Our target distribution is $p^*(x_{1:T,1:K}) \stackrel{\text{def}}{=} p(x_{1:T,1:K} \mid y_{1:T})$. We can write this as

$$p^*(x) \propto \prod_{t=1}^{T} \left[\prod_{k=1}^{K} p(x_{t,k} \mid x_{t-1,k}) \times p(y_t \mid x_{t,1:K}) \right]. \quad (16)$$

$\mathcal{F}$ handles the transition potentials $p(x_{t,k} \mid x_{t-1,k})$ exactly, but approximates the observation potential $p(y_t \mid x_{t,1:K})$ by a factored potential $\prod_{k=1}^{K} \hat{\psi}_{t,k}(x_{t,k})$. Our unnormalized approximating distribution $\hat{f}_{\theta_a}$ over a° is therefore

$$\hat{f}_{\theta_a}(x) = \prod_{t=1}^{T} \left[\prod_{k=1}^{K} p(x_{t,k} \mid x_{t-1,k}) \times \prod_{k=1}^{K} \hat{\psi}_{t,k}(x_{t,k}) \right]. \quad (17)$$

We use the following formula for $\hat{\psi}_{t,1:K}$, which corresponds to a single forward pass of expectation propagation (Minka, 2001):

$$\hat{\psi}_{t,k} = \begin{cases} \delta(x_{t,k} = y_t) & : \ t \geq t_0, \hat{c}_t = k \\ 1 & : \ t \geq t_0, \hat{c}_t \neq k \\ \frac{\delta(x_{t,k}=y_t) + \sum_{l \neq k} p_{\text{prior}}(x_{t,l}=y_t)}{K} & : \ t < t_0. \end{cases} \quad (18)$$

Here $p_{\text{prior}}(x_{t,l} = y_t)$ is the marginal probability that $x_{t,l}$ is equal to y_t under the prior.

The observations $y_{1:(t_0-1)}$ are incorporated approximately via the factored variational approximation, while the observations $y_{t_0:T}$ are incorporated exactly. We therefore obtain the ability to exactly incorporate recent information that may be most relevant to predicting y_T while still benefiting from information further in the past.

4. Algorithms for Optimizing the Partition

Recall the framework introduced in Section 2; given a hierarchical decomposition A, we choose a partition by searching over subsets $\{\mathcal{X}\} \subseteq B \subseteq A$, attempting to optimize the objective given in (8).

In this section, we present heuristics for solving this optimization problem. To start, we will assume that A is small enough for algorithms that are polynomial-time in $|A|$ to be feasible; in this case we present an $O(|A| \log |A|)$ greedy algorithm. (There is also a potentially more accurate $O(|A|dk^2)$ dynamic programming algorithm, described in the supplementary material, which exactly optimizes a surrogate for $\text{KL}(p^* \parallel \hat{p})$.)

Then, we will consider the case where $|A|$ is too large for even a linear-time algorithm to be feasible, and introduce an analogue of beam search; this is the form of the algorithm that we use in our experiments.

Greedy heuristic. We define the *local probability mass* of each region a to be the unnormalized mass that $\hat{f}_{\theta_a}$ assigns to a°: $m_{\text{loc}}(a) \stackrel{\text{def}}{=} \sum_{x \in a^\circ} \hat{f}_{\theta_a}(x)$. Then we simply greedily include the k elements of A where $m_{\text{loc}}(a)$ is the largest, together with $\mathcal{X}$. This only requires sorting by m_{loc} and so has runtime $O(|A| \log |A|)$.

Abstract beam search. In both of the examples in Section 3, the hierarchical decomposition A was exponentially large, so we adopt an incremental refinement strategy: write the target distribution $p^*(x)$ as an unnormalized product of potentials $p^*(x) \propto \prod_{t=1}^{T} \psi_t(x)$, and let $A_0 = \{\mathcal{X}\}$. Then, for $t = 1, \ldots, T$ execute the following sequence of steps:

1. **Refine** A_{t-1} to get $\tilde{A}_t$.
2. **Prune** $\tilde{A}_t$ down to a subset A_t of size k by solving (8) with target distribution $\prod_{s=1}^{t} \psi_s(x)$.

For the pruning step, we can use the greedy algorithm discussed above. For the refinement step, we need to make sure that $\tilde{A}_t$ is a hierarchical decomposition whenever A_{t-1} is. One way to do this is to set $\tilde{A}_t = \{\mathcal{X}\} \cup \bigcup_{a \in A_{t-1}} r_t(a)$, where r_t is a *refinement operator* satisfying the following properties: (i) $r_t(a)$ is a partition of a; (ii) if $b \subsetneq a$, then every element of $r_t(b)$ is contained in an element of $r_t(a)$.

It is also possible to construct refinement operators satisfying weaker conditions, but the conditions above will suffice for our purposes.

Example 4.1. In filtering, where $\mathcal{X} = \Sigma^T$, a natural choice for $\psi_t(x)$ is $p(x_t \mid x_{1:t-1}, y_t)$, and a natural choice for r_t is to partition based on the value of x_t. This is illustrated

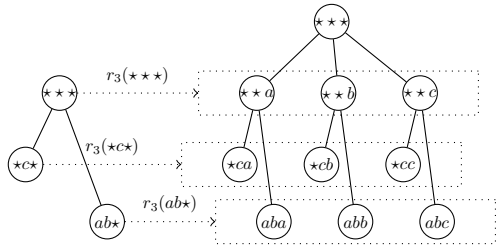

Figure 3. A hierarchical decomposition A_2 (left) together with its refinement $\tilde{A}_3$ (right). These correspond to the refinement strategy in Example 4.1. Each dotted rectangle indicates the refinements of a particular region in A_2.

Algorithm 1 Abstract beam search algorithm. Inputs are the space $\mathcal{X}$, a refinement function r, a fitting method `Fit`, and a beam size k. Generates a sequence of hierarchical decompositions, each of which defines a distribution over $\mathcal{X}$.

AbstractBeamSearch($\mathcal{X}$, r, `Fit`, k)
$A_0 = \{\mathcal{X}\}$
for $t = 1$ **to** T **do**
  $\tilde{A}_t = \{\mathcal{X}\} \cup \bigcup_{a \in A_{t-1}} r_t(a)$
  for $a \in \tilde{A}_t$ **do**
    $\theta_a =$ result of using `Fit` with target $\prod_{s=1}^{t} \psi_s$
    $m_{\text{loc}}(a) = \hat{f}_{\theta_a}(a) - \sum_{b \in \mathcal{C}_{\tilde{A}_t}(a)} \hat{f}_{\theta_a}(b)$
  end for
  $A_t = \{\mathcal{X}\} \cup \{k$ items in $\tilde{A}_t$ with largest $m_{\text{loc}}\}$
end for

in Figure 3, where a refinement function r_3 is used to transition from A_2 to $\tilde{A}_3$. It is also the choice of r_t used for the n-gram task in Section 5.

Summary. Algorithm 1 depicts the pseudocode for our abstract beam search algorithm. The major steps at each stage of the algorithm are to refine the particles from step $t-1$, then to update the approximation to p^* based on a new potential ψ_t, then to compute the local probability mass $m_{\text{loc}}(a)$ for each region a using (10), and finally to prune the hierarchical decomposition down to size k.

5. Experiments

To test our approach, we performed experiments on both of the filtering examples in Section 3.

For the text reconstruction task, we fit an n-gram model for the transitions using interpolated Kneser-Ney (Kneser & Ney, 1995) trained on *The Complete Works of William Shakespeare* (about $125,000$ lines in total). The first $115,000$ lines were used to train the model and each of the next $5,000$ lines were used as a development and test set, respectively. Interpolated Kneser-Ney has two hyper-parameters: the model order n and the discount parameter λ. We fit these by minimizing perplexity on the development set, and found that $n = 8$, $\lambda = 0.9$ was optimal. In the test set, 75% of the characters were randomly replaced with a wildcard symbol. The inference task was to recover the original value of each of the replaced characters.

For the multiple object tracking task, we generated synthetic data. Each of K objects labeled $1, \ldots, K$ travel around a circle and each second one of the objects (but not its label) is observed. The inference task is to recover the label of the observed objects (their initial positions are known).

Task characteristics and particle coverage. We now study the characteristics of our tasks. Our intuition was that the text reconstruction task is "easy" while the object-tracking task is "hard". This is because, even though there are many *possible* 8-grams, the number of *plausible* 8-grams is not too large, since there are strong correlations between one character and the next. In contrast, the number of plausible states for the object-tracking task grows exponentially with the number of objects. To validate this intuition, we performed a preliminary experiment to determine the effective size of the state space for both tasks. We performed beam search under the prior, then plotted the amount of probability mass covered by the top k elements of the beam. The point of this was to determine how concentrated the probability mass is under the model (a beam size of $30,000$ was used in order to capture all high-probability regions). The results are given in Figure 4. For the text reconstruction task, most of the mass is covered by relatively few elements; in contrast, for the object-tracking task, even thousands of elements are insufficient to provide coverage.

Quality of particles. Next, we wanted to test the hypothesis that abstract particles generate a higher quality approximate posterior than concrete particles. We used prediction accuracy as our scoring metric (i.e. a score of 1 if the posterior mode for an example is at the truth and 0 otherwise).

The results of our experiments are given in Figures 5 and 6. In Figure 5, we plotted accuracy versus number of particles. For the text reconstruction task, abstract beam search outperforms concrete beam search, which slightly outperforms SMC. For the object tracking task, abstract beam search again outperforms both other methods, but this time concrete beam search performs poorly compared to SMC.

Measuring computation. We have demonstrated that abstract particles are higher quality per-particle, but this doesn't address their additional overhead. This is especially important for the object-tracking task, where each abstract particle requires a forward pass of HMM inference

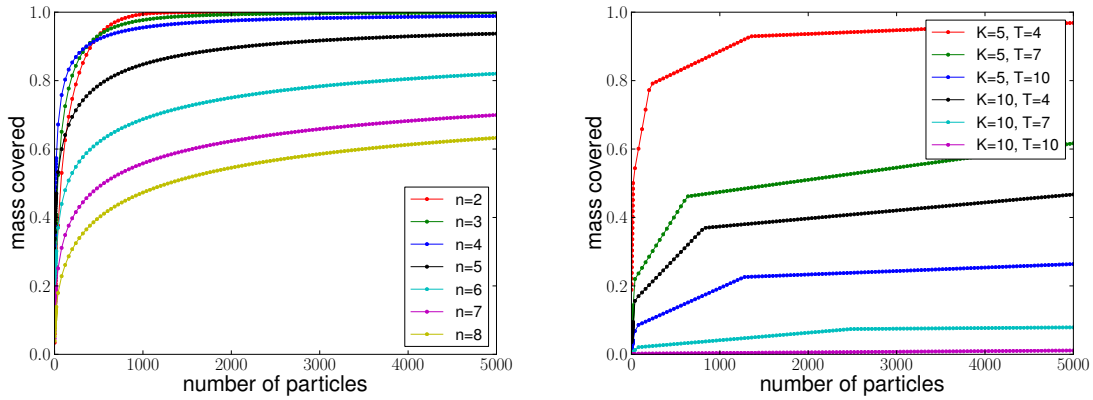

Figure 4. Left: coverage of the prior distribution over the first n characters for text reconstruction (n-gram prior). Right: coverage of the prior distribution over the first T seconds for object-tracking (factorial HMM prior), for $T \in \{4, 7, 10\}$. Note that some curves asymptote to less than 1; this happens when a non-negligible amount of probability mass falls off the beam.

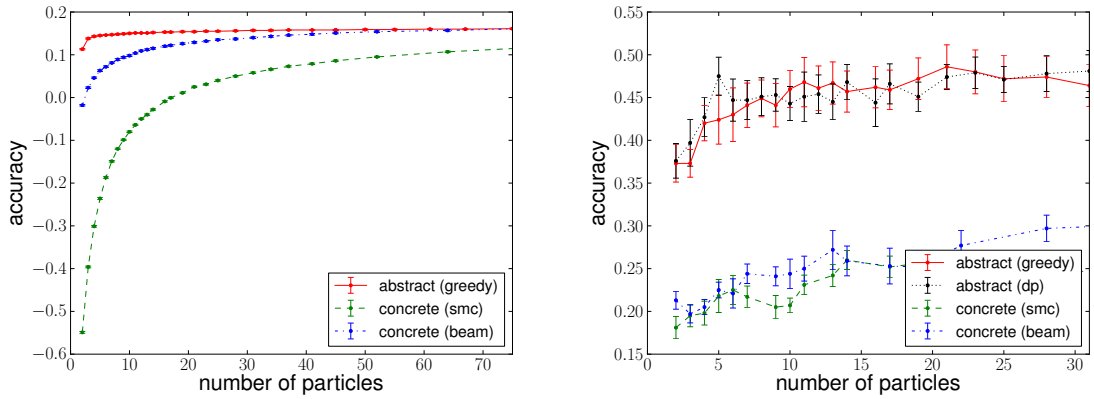

Figure 5. Number of particles (k) vs. accuracy, for abstract beam search, beam search, and SMC. Left: text reconstruction with n-gram prior, for $n = 8$; right: object-tracking task for 15 objects with state size 100.

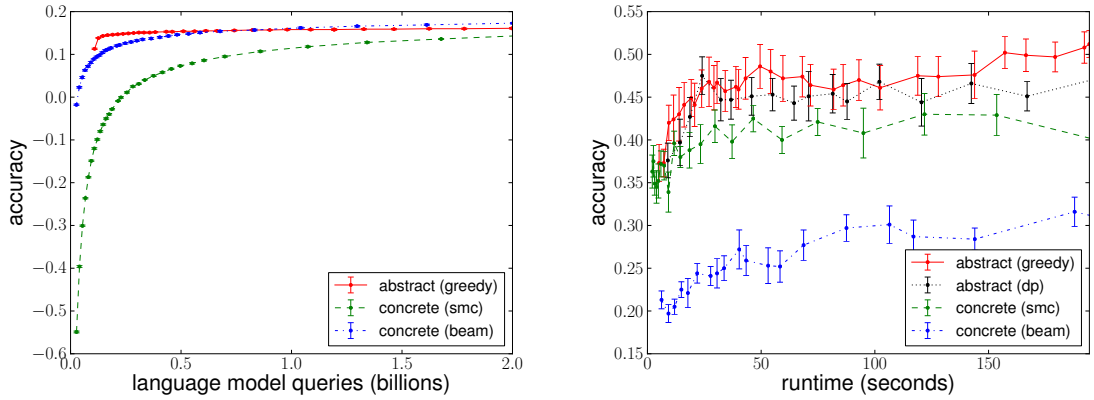

Figure 6. Computational cost vs. accuracy, for abstract beam search, concrete beam search, and SMC. The cost metric is language model queries for the text reconstruction task and runtime for the object-tracking task. Runtime was computed using a single core of a 3.4GHz machine with 32GB of RAM. These plots correspond to those of Figure 5.

and thus requires memory proportional to $K \cdot S$. We therefore also plotted computational costs against accuracy for both models in Figure 6. For the text reconstruction task, we used the number of queries to the language model to measure cost, as this is a bottleneck in many applications. We found that despite abstract beam search requiring more queries per particle, the same story holds as before: abstract beam search outperforms beam search, which outperforms SMC. The exception is when the number of queries is large, in which case SMC slightly outperforms our algorithm.

For the object-tracking task, we used runtime as our metric, and also included a variant of our algorithm where dynamic programming is used in place of the greedy algorithm during the pruning step. Again, the same story as before holds: abstract beam search outperforms both concrete beam search and SMC. Furthermore, we found that the greedy pruning heuristic was typically comparable to or better than pruning with dynamic programming. This suggests that we can use the greedy pruner without losing accuracy.

Details of SMC sampler. We provide here the details of our (concrete) SMC implementation to provide more context to our comparisons. In both tasks, we used $p(x_{t+1} \mid x_t, y_{t+1})$ as the proposal distribution. We performed resampling after each step since this was not a performance bottleneck. For the object-tracking task, one major issue was particle collapse, wherein all of the particles received zero mass (due to being incompatible with the observations). When this happened, we "re-initialized" the particles by sampling from the prior marginal and then randomly changing the location of one of the objects for each particle to match the current observation. We found that this heuristic substantially improved the performance of SMC on the object-tracking task.

6. Discussion

We have presented a new "abstract" filtering algorithm based on hierarchical decompositions, which combines the flexibility of particle-based algorithms (e.g., SMC) with the compactness of parametrized approximations (e.g., variational inference). Broadly speaking, our approach shares similar intuitions as many other coarse-to-fine methods such as decision and density estimation trees (Ram & Gray, 2011), state-split grammars (Petrov et al., 2006), structured prediction cascades (Weiss & Taskar, 2010), and clustering variables in Markov logic (Kiddon & Domingos, 2011).

The work most closely related to ours is the split variational inference framework of (Bouchard & Zoeter, 2009). Split variational inference uses a similar decomposition to that given in Section 2.1, but uses variational Bayes to fit the parameters. They also use soft partitions (non-negative functions summing to 1) given by a continuously parameterized family, allowing numerical optimization of the partitions in contrast to our combinatorial approach in Section 2.2. In this work we also make use of a beam search heuristic in conjunction with variational approximation, which allows us to scale our approach to difficult combinatorial inference problems. It would be interesting to determine whether the idea of smooth partitions could be used to extend our ideas to continuous state spaces.

Another line of related work is logical particle filtering (Zettlemoyer et al., 2007; Hajishirzi & Amir, 2012), which performs sequential Monte Carlo over first order predicates. This approach shares our goal of obtaining compact representations for inference, but focuses on obtaining an exact representation of the posterior within each region, and also does not use a collection of regions that covers the space. Logical predicates are a way to specify rich but flexible hierarchical decompositions, and so would be interesting to apply in our context.

Other related work includes quasi-Monte Carlo, which uses deterministic sample placement to speed up Markov chain convergence (Niederreiter, 1992; Caflisch, 1998); and various hierarchical Bayesian nonparametric techniques (Heller & Ghahramani, 2005; Gramacy & Lee, 2008; Fox & Dunson, 2012), which often use variational techniques to infer a hierarchy.

Our work can also be viewed as performing a form of variational inference with feature induction in log-linear models (Pietra et al., 1997; McCallum, 2003), where base features are conjoined with flexible features corresponding to regions of the input space. This view strengthens the connection between SMC and variational inference, and we believe this bridge will allow us to develop new families of inference algorithms.

Acknowledgments Thanks to the anonymous reviewers, whose comments substantially improved the paper. The first author was supported by the Hertz Foundation.

References

Bouchard, Guillaume and Zoeter, Onno. Split variational inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 57–64. ACM, 2009.

Bouchard-Côté, Alexandre, Sankararaman, Sriram, and Jordan, Michael I. Phylogenetic inference via sequential monte carlo. *Systematic biology*, 61(4):579–593, 2012.

Caflisch, Russel E. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 1998:1–49, 1998.

Cappé, Olivier, Godsill, Simon J, and Moulines, Eric. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

Carreras, Xavier and Collins, Michael. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 200–209. Association for Computational Linguistics, 2009.

Doucet, Arnaud and Johansen, Adam M. A tutorial on particle filtering and smoothing: Fifteen years later. In *Oxford Handbook of Nonlinear Filtering*. Citeseer, 2011.

Fox, Emily B and Dunson, David B. Multiresolution gaussian processes. *arXiv preprint arXiv:1209.0833*, 2012.

Gilks, Walter R and Berzuini, Carlo. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

Görür, Dilan and Teh, Yee W. An efficient sequential monte carlo algorithm for coalescent clustering. In *Advances in Neural Information Processing Systems*, pp. 521–528, 2008.

Gramacy, Robert B and Lee, Herbert KH. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483), 2008.

Hajishirzi, Hannaneh and Amir, Eyal. Sampling first order logical particles. *arXiv preprint arXiv:1206.3264*, 2012.

Heller, Katherine A and Ghahramani, Zoubin. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pp. 297–304. ACM, 2005.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, and Saul, Lawrence K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kešelj, Vlado, Peng, Fuchun, Cercone, Nick, and Thomas, Calvin. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pp. 255–264, 2003.

Kiddon, C. and Domingos, P. Coarse-to-fine inference and learning for first-order probabilistic models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.

Klein, Dan, Smarr, Joseph, Nguyen, Huy, and Manning, Christopher D. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 180–183. Association for Computational Linguistics, 2003.

Kneser, Reinhard and Ney, Hermann. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pp. 181–184. IEEE, 1995.

Liang, P., Jordan, M. I., and Klein, D. Learning dependency-based compositional semantics. In *Association for Computational Linguistics*, pp. 590–599, 2011.

McCallum, A. Efficiently inducing features of conditional random fields. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 403–410, 2003.

Minka, Thomas P. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

Niederreiter, Harald. *Quasi-Monte Carlo Methods*. Wiley Online Library, 1992.

Pal, Chris, Sutton, Charles, and McCallum, Andrew. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pp. V–V. IEEE, 2006.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. Learning accurate, compact, and interpretable tree annotation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, pp. 433–440, 2006.

Pietra, S. D., Pietra, V. D., and Lafferty, J. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Quirk, Chris and Moore, R. Faster beam-search decoding for phrasal statistical machine translation. *Machine Translation Summit XI*, 2007.

Ram, P. and Gray, A. G. Density estimation trees. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 627–635, 2011.

Vatanen, Tommi, Väyrynen, Jaakko J, and Virpioja, Sami. Language identification of short text segments with n-gram models. In *LREC*, 2010.

Weiss, D. and Taskar, B. Structured prediction cascades. In *Artificial Intelligence and Statistics (AISTATS)*, 2010.

Zettlemoyer, Luke S, Pasula, Hanna M, and Kaelbling, Leslie Pack. Logical particle filtering. In *Probabilistic, Logical and Relational Learning-A Further Synthesis*, 2007.