Message Passing Algorithms for Dirichlet Diffusion Trees

David A. Knowles, University of Cambridge
Jurgen Van Gael, Microsoft Research Cambridge
Zoubin Ghahramani, University of Cambridge

May 6, 2011

Abstract

We demonstrate efficient approximate inference for the Dirichlet Diffusion Tree (Neal, 2003), a Bayesian nonparametric prior over tree structures. Although DDTs provide a powerful and elegant approach for modeling hierarchies they haven't seen much use to date. One problem is the computational cost of MCMC inference. We provide the first deterministic approximate inference methods for DDT models and show excellent performance compared to the MCMC alternative. We present message passing algorithms to approximate the Bayesian model evidence for a specific tree. This is used to drive sequential tree building and greedy search to find optimal tree structures, corresponding to hierarchical clusterings of the data. We demonstrate appropriate observation models for continuous and binary data. The empirical performance of our method is very close to the computationally expensive MCMC alternative on a density estimation problem, and significantly outperforms kernel density estimators.

1 Introduction

Tree structures play an important role in machine learning and statistics. Learning a tree structure over data points gives a straightforward picture of how objects of interest are related. Trees are easily interpreted and intuitive to understand. Sometimes we may know that there is a true underlying hierarchy: for example species in the tree of life or duplicates of genes in the human genome, known as paralogs. Typical mixture models, such as Dirichlet Process mixture models, have independent parameters for each component. In many situations we might expect that certain clusters are similar, for example are sub-groups of some large group. By learning this hierarchical similarity structure, the model can share statistical strength between components to make better estimates of parameters using less data.

In Bayesian modelling there is also a very good statistical reason to use tree structured prior distributions. By coupling priors through a tree structure, we allow posterior information to flow between the branches in the tree. This sharing of statistical strength is often crucial for meaningful analysis in contexts where little data is available.

Traditionally, a family of methods known as *hierarchical clustering* is used to learn tree structures. Classical hierarchical clustering algorithms employ a bottom up "agglomerative" approach (Duda et al., 2001): start with a "hierarchy" of one datapoint and iteratively add the closest datapoint in some metric into the hierarchy. The output of these algorithms is usually a binary tree or dendrogram. Although these algorithms are straightforward to implement, both the ad-hoc method and the distance metric hide the statistical assumptions being made.

Two solutions to this problem have recently been proposed. In Heller & Ghahramani the bottom-up agglomerative approach is kept but a principled probabilistic model is used to find subtrees of the hierarchy. Bayesian evidence is then used as the metric to decide which node to incorporate in the tree. Although fast, the lack of a generative process prohibits modeling uncertainty over tree structures. A different line of work (Williams, 2000; Neal, 2003; Teh et al., 2008; Blei et al., 2010; Roy et al., 2006) starts from a generative probabilistic model for both the tree structure and the data. Bayesian inference machinery can then be used to compute posterior distributions on both the internal nodes of the tree as well as the tree structures themselves.

An advantage of the generative probabilistic models for trees is that they can be used as a building block for other latent variable models (Rai & Daumé III, 2008). We could use this technique to build topic models with hierarchies on the topics, or hidden Markov models where the states are hierarchically related. Greedy agglomerative approaches can only cluster latent variables *after* inference has been done and hence they cannot be used in a principled way to aid inference in the latent variable model.

In this work we use the Dirichlet Diffusion Tree (DDT) introduced in Neal (2003), and reviewed in Section 2. This simple yet powerful generative model specifies a distribution on binary trees with multivariate Gaussian distributed variables at the leaves. The DDT is a Bayesian nonparametric prior, and is a generalization of Dirichlet Process mixture models (Rasmussen, 2000). The DDT can be thought of as providing a very flexible density model, since the hierarchical structure is able to effectively fit non-Gaussian distributions. Indeed, in Adams et al. (2008) the DDT was shown to significantly outperform a Dirichlet Process mixture model in terms of predictive performance, and in fact slightly outperformed the Gaussian Process Density Sampler. The DDT is thus both a mathematically elegant nonparametric distribution over hierarchies and provides state-of-the-art density estimation performance.

Our algorithms use the message passing framework (Kschischang et al., 2001; Minka, 2005). For many models message passing has been shown to significantly outperform sampling methods in terms of speed-accuracy trade-off. However, general α-divergence (Minka, 2005) based message passing is not guaranteed to

converge, which motivates our second, guaranteed convergent, algorithm which uses message passing within EM (Kim & Ghahramani, 2006). Secondly, it is inherently parallelizable which is increasingly important on modern computer architectures. Thirdly, message passing algorithms are modular: the message passing rules for the DDT only need to be implemented once and can then easily be combined with message passing on various latent variable model. Both algorithms are deterministic which confers various advantages over MCMC methods. Running inference on the same model and same data is reproducible. Convergence is simple compared to assessing convergence of a Markov chain.

The contributions of this paper are as follows. We derive and demonstrate full message passing (Section 3.1) and message passing within EM algorithms (Section 3.2) to approximate the model evidence for a specific tree, including integrating over hyperparameters (Section 3.3). We show how the resulting approximate model evidence can be used to drive greedy search over tree structures (Section 3.4). We demonstrate that it is straightforward to connect different observation models to this module to model different data types, using binary vectors as an example. Finally we present experiments using the DDT and our approximate inference scheme in Section 4.

2 The Dirichlet Diffusion Tree

The Dirichlet Diffusion Tree was introduced in Neal (2003) as a top-down generative model for trees over N datapoints $x_1, x_2, \cdots, x_N \in \mathcal{R}^D$. The dataset is generated sequentially with each datapoint x_i following the path of a Brownian motion for unit time. We overload our notation by referring to the position of each datapoint at time t as $x_i(t)$ and define $x_i(1) = x_i$.

The first datapoint starts at time 0 at the origin in a D-dimensional Euclidean space and follows a Brownian motion with variance σ^2 until time 1. If datapoint 1 is at position $x_1(t)$ at time t, the point will reach position $x_1(t + dt) = x_1(t) + \text{Normal}(0, \sigma^2 dt)$ at time $t + dt$. It can easily be shown that $x_1(t) \sim \text{Normal}(0, \sigma^2 t)$. The second point x_2 in the dataset also starts at the origin and initially follows the path of x_1. The path of x_2 will diverge from that of x_1 at some time T_d (controlled by the "divergence function" $a(t)$) after which x_2 follows a Brownian motion independent of $x_1(t)$ until $t = 1$. In other words, the infinitesimal increments for the second path are equal to the infinitesimal increments for the first path for all $t < T_d$. After T_d, the increments for the second path $\text{Normal}(0, \sigma^2 dt)$ are independent.

The generative process for datapoint i is as follows. Initially $x_i(t)$ follows the path of the previous datapoints. If x_i does not diverge before reaching a previous branching point, the previous branches are chosen with probability proportional to how many times each branch has been followed before. This reinforcement scheme is similar to the Chinese restaurant process (Aldous, 1985). If at time t the path for datapoint i has not yet diverged, it will diverge in the next infinitesimal time step dt with probability $a(t)dt/m$, where m is the number of datapoints that have previous followed the current path. The division by m is

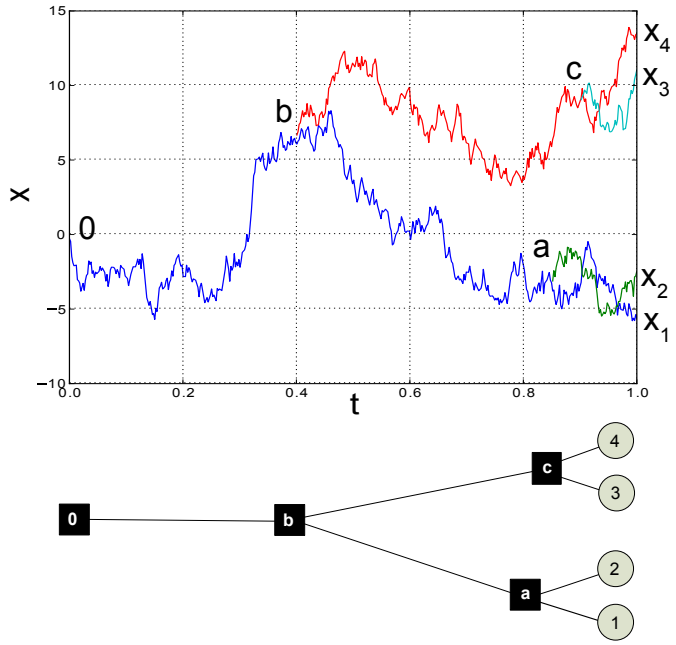

Figure 1: A sample from the Dirichlet Diffusion Tree with $N = 4$ datapoints. Top: the location of the Brownian motion for each of the four paths. Bottom: the corresponding tree structure. Each branch point corresponds to an internal tree node.

another reinforcing aspect of the DDT: the more datapoints follow a particular branch, the more likely subsequent datapoints will not diverge off this branch.

For the purpose of this paper we use the divergence function $a(t) = \frac{c}{1-t}$, with "smoothness" parameter $c > 0$. Larger values $c > 1$ give smoother densities because divergences typically occur earlier, resulting in less dependence between the datapoints. Smaller values $c < 1$ give rougher more "clumpy" densities with more local structure since divergence typically occurs later, closer to $t = 1$. We refer to Neal (2001) for further discussion of the properties of this and other divergence functions. Figure 1 illustrates the diffusion tree process for a dataset with $N = 4$ datapoints.

The probability of generating a dataset under the DDT can be decomposed into two components. The first component specifies the distribution over the tree structure and the divergence times. The second component specifies the distribution over the specific locations of the Brownian motion when the tree structure and divergence times are given.

Before we describe the functional form of the DDT prior we will need two results. First, the probability that a new path does not diverge between times $s < t$ on a segment that has been followed m times by previous data-points can

be written as

$$P(\text{not diverging}) = \exp\left[(A(s) - A(t))/m\right],$$

where $A(t) = \int_0^t a(u)du$ is the cumulative rate function. For our divergence function $A(t) = -c\log(1-t)$. Second, the DDT prior defines an exchangeable distribution: the order in which the datapoints were generated does not change the joint density. See Neal (2003) for a proof.

We now consider the tree as a set of segments $\mathcal{S}(\mathcal{T})$ each contributing to the joint probability density. The tree structure $\mathcal{T}$ contains the counts of how many datapoints traversed each segment. Consider an arbitrary segment $[ab] \in \mathcal{S}(\mathcal{T})$ from node a to node b with corresponding locations x_a and x_b and divergence times t_a and t_b. Let $m(b)$ be the number of leaves under node b, i.e. the number of datapoints which traversed segment $[ab]$. Let $l(b)$ and $r(b)$ be the number of leaves under the left and right child of node b respectively, so that $l(b) + r(b) = m(b)$.

By exchangeability we can assume that it was the second path which diverged at b. None of the subsequent paths diverged before time t_b (otherwise $[ab]$ would not be a contiguous segment). The probability of this happening is

$$\begin{aligned} P(t_b|[ab], t_a) &= a(t_b) \prod_{i=1}^{m(b)-1} \exp[(A(t_a) - A(t_b))/i] \\ &= a(t_b) \exp\left[(A(t_a) - A(t_b))H_{m(b)-1}\right] \end{aligned}$$

where $H_n = \sum_{i=1}^{n} 1/i$ is the nth harmonic number. This expression factorizes into a term for t_a and t_b. Collecting such terms from the branches attached to an internal node i the factor for t_i for the divergence function $a(t) = c/(1-t)$ is

$$\begin{aligned} & a(t_i)e^{[A(t_i)(H_{l(i)-1} + H_{r(i)-1} - H_{m(i)-1})]} \\ & = (1-t_i)^{cJ_{l(i),r(i)}-1} \end{aligned} \tag{1}$$

where $J_{l,r} = H_{r+l-1} - H_{l-1} - H_{r-1}$.

Each path that went through x_b, except the first and second, had to choose to follow the left or right branch. Again, by exchangeability, we can assume that all $l(b) - 1$ paths took the left branch first, then all $r(b) - 1$ paths chose the right branch. The probability of this happening is

$$P([ab]) = \frac{(l(b)-1)!(r(b)-1)!}{m(b)!}$$

Finally, we include a term for the diffusion locations:

$$P(x_b|x_a, t_a, t_b) = \text{Normal}(x_b; x_a, \sigma^2(t_b - t_a)) \tag{2}$$

The full joint probability for the DDT is now a product of terms for each segment

$$P(x, t, \mathcal{T}) = \prod_{[ab]\in\mathcal{S}(\mathcal{T})} P(x_b|x_a, t_a, t_b)P(t_b|[ab], t_a)P([ab])$$

3 Approximate Inference for the DDT

We assume that the likelihood can be written as a product of conditional probabilities for each of the leaves x_n: $\prod_n l(y_n|x_n)$ where y_n is observed data. Our aim is to calculate the posterior distribution

$$P(x,t,\mathcal{T}|y) = \frac{P(y,x,t,\mathcal{T})}{\sum_{\mathcal{T}} \int P(y,x,t,\mathcal{T}) dx dt}$$

Unfortunately, this integral is analytically intractable. Our solution is to use message passing or message passing within EM to approximate the marginal likelihood for a given tree structure: $P(y|\mathcal{T}) = \int P(y,x,t|\mathcal{T}) dx dt$. We use this approximate marginal likelihood to drive tree building/search algorithm to find a weighted set of K-best trees.

3.1 Message passing algorithm

Here we describe our message passing algorithm for a fixed tree structure $\mathcal{T}$. We employ the α-divergence framework from Minka (2005). Our variational approximation is fully factorized with a Gaussian q for every variable except c (the divergence function parameter) which is Gamma distributed. For each segment $[ab] \in \mathcal{S}(\mathcal{T})$ we introduce two variables which are deterministic functions of existing variables: the branch length $\Delta_{[ab]} = t_b - t_a$ and the variance $v_{[ab]} = \sigma^2 \Delta_{[ab]}$. We now write the unnormalized posterior as a product of factors:

$$\begin{aligned} &\prod_{n \in \text{leaves}} l(y_n|x_n) \prod_{[ab] \in \mathcal{S}(\mathcal{T})} N(x_b; x_a, v_{[ab]}) \\ &\times \delta(v_{[ab]} - \sigma^2 \Delta_{[ab]}) \delta(\Delta_{[ab]} - (t_b - t_a)) \\ &\times \mathcal{I}(0 < \Delta_{[ab]} < 1) P(t_b|\mathcal{T}) \end{aligned} \tag{3}$$

where $\delta(.)$ is the Dirac delta spike at 0, and $\mathcal{I}(.)$ is the indicator function. These functions are used to break down more complex factors into simpler ones for computational and mathematical convenience. Equation 3 defines a factor graph over the variables, shown in Figure 2.

Choosing α-divergences. We choose an α for each factor f in the factor graph and then minimize the α-divergence $D_\alpha[q^{\sim f}(W)\tilde{f}(W)||q^{\sim f}(W)f(W)]$ with respect to $\tilde{f}(W)$ where $W = \{x, t, \Delta, v\}$ is the set of latent variables for all nodes. Here

$$D_\alpha(p||q) = \frac{1}{\alpha(1-\alpha)} \int 1 - p(x)^\alpha q(x)^{(1-\alpha)} dx$$

is the α-divergence between two (normalized) distributions p and q; and $q^{\sim f}(W) = q(W)/\tilde{f}(W)$ is the cavity distribution: the current variational posterior without the contribution of factor f. Minka (2005) describes how this optimization can be implemented as a message passing algorithm on the factor graph.

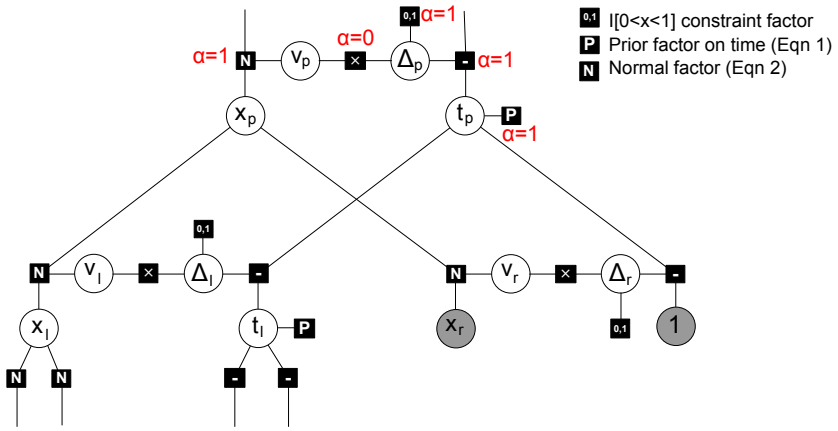

Figure 2: A subsection of the factor graph for the Dirichlet Diffusion Tree. The left node represents an internal node and is connected to two child nodes (not depicted). The right node is a leaf node hence its location and divergence time are observed (denoted in gray). The choice of α for the factors is shown for the parent node. The hyperparameters σ^2 and c which are connected to the $\times$ (multiplication) factor and time prior P respectively are not shown.

We choose which α-divergence to minimize for each factor considering performance and computational tractability. The normal, minus, divergence time prior and constraint factors use $\alpha = 1$. The multiplication factor and prior on divergence function parameter c use $\alpha = 0$ (Figure 2). For the normal factor we use $\alpha = 1$ to attempt to approximate the evidence unbiasedly ($\alpha = 0$ only lower bounds the evidence, see Minka (2005)). The divergence time prior is nonconjugate so it is more straightforward to use $\alpha = 0$ (EP). For the constraint factors the divergence is infinite for $\alpha = 0$ but analytic for $\alpha = 1$. For the multiplication factor we use $\alpha = 0$ due to the multimodal posterior (Stern et al., 2009). Since we only use $\alpha = 0$ and 1 our algorithm can also be viewed as a hybrid Expectation Propagation (Minka, 2001) and Variational Message Passing (Winn & Bishop, 2006)/mean field (Beal, 2003) algorithm. We use the Infer.NET (Minka et al., 2010) low level library of message updates to calculate the outgoing message from each factor.

Further approximations. We found several approximations to the full message passing solution to be beneficial to the accuracy-speed trade-off of our algorithm.

- The message from the divergence time prior is a Beta distribution. Calculating the true EP message when $q(t)$ is Gaussian would require quadrature, which we found to be less accurate and more computationally expensive than the following: map the incoming Gaussian message to a Beta distribution with the same mean and variance; multiply in the Beta

message; then map the outgoing message back to a Gaussian, again by matching moments.

- For practically sized trees (i.e. with 10 or more leaves) we found the message from the variance to the normal factor was typically quite peaked. We found no significant loss in performance in using only the mean of this message when updating the location marginals. In fact, since this removes the need to do any quadrature, we often found the performance was improved.

- Similarly for the divergence function parameter c, we simply use the mean of the incoming message, since this was typically quite peaked.

Approximating the model evidence is required to drive the search over tree structures (see Section 3.4). Our evidence calculations follow Minka (2005), to which we defer for details. We use the evidence calculated at each iteration to assess convergence.

Scheduling. Sensible scheduling of the message passing algorithm aided performance. The factor graph consists of two trees: one for the divergence locations, one for the divergences times, with the branch lengths and variances forming cross links between the trees. Belief propagation on a tree is exact: in one sweep up and then down the tree all the marginals are found. Although this is not the case when the divergence times are unknown or KL projection is required at the leaves, it implies that such sweeps will propagate information efficiently throughout the tree, since EP is closely related to BP. To propagate information efficiently throughout this factor graph, our schedule consists of one sweep up and down the tree of locations and tree of times, followed by one sweep back and forth along the cross-links.

Usually one would start with the messages coming in from the prior: for example for the divergence times. Unfortunately in our case these messages are improper, and only result in proper marginals when the constraints that $t_a < t_b$ are enforced through the constraint that the branch length $\Delta_{a\to b}$ must be positive. To alleviate this problem we initially set the message from the prior to spread the divergence times in the correct order, between 0 and 1, then run an iteration of message passing on the tree of divergence times, the constraints $0 < \Delta_{a\to b} < 1$ and $0 < t < 1$ and the prior. This results in proper variance messages into the Normal factors when we sweep over the tree of location times.

3.2 Message passing in EM algorithm

For high dimensional problems we have found that our message passing algorithm over the divergence times can have convergence problems. This can be addressed using damping, or by maximizing over the divergence times rather than trying to marginalize them. In high dimensional problems the divergence times tend to have more peaked posteriors because each dimension provides independent information on when the divergence times should be. Because of this,

and because of the increasing evidence contribution from the increasing number of Gaussian factors in the model at higher dimension D, modeling the uncertainty in the divergence times becomes less important. This suggests optimizing the divergence times in an EM type algorithm.

In the E-step, we use message passing to integrate over the locations and hyperparameters. In the M-step we maximize the lower bound on the marginal likelihood with respect to the divergence times. While directly maximizing the marginal likelihood itself is in principle possible calculating the gradient with respect to the divergence times is intractable since all the terms are all coupled through the tree of locations. For each node i with divergence time t_i we have the constraints $t_p < t_i < \min(t_l, t_r)$ where t_l, t_r, t_p are the divergence times of the left child, right child and parent of i respectively.

One simple approach is to optimize each divergence time in turn (e.g. using golden section search), performing a co-ordinate ascent. However, we found jointly optimizing the divergence times using LBFGS (Liu & Nocedal, 1989) to be more computationally efficient. Since the divergence times must lie within $[0, 1]$ we use the reparameterization $s_i = \log[t_i/(1-t_i)]$ to extend the domain to the whole space, which we find improves empirical performance. From Equations 2 and 1 the lower bound on the log evidence with respect to an individual divergence time t_i is

$$
\begin{aligned}
&(\langle c \rangle J_{l(i),r(i)} - 1)\log(1-t_i) - a\log(t_i - t_p) - \langle \frac{1}{\sigma^2} \rangle \frac{b_{[pi]}}{t_i - t_p} \\
&a = \frac{D}{2}, \qquad b_{[pi]} = \frac{1}{2}\sum_{d=1}^{D} \mathbb{E}[(x_{di} - x_{dp})^2]
\end{aligned} \tag{4}
$$

where x_{di} is the location of node i in dimension d, and p is the parent of node i. The full lower bound is the sum of such terms over all nodes. The expectation required for $b_{[pi]}$ is readily calculated from the marginals of the locations after message passing. Differentiating to obtain the gradient with respect to t_i is straightforward so we omit the details. The chain rule is used to obtain the gradient with respect to the reparameterisation s_i, i.e. $\frac{\partial \cdot}{\partial s_i} = \frac{\partial t_i}{\partial s_i}\frac{\partial \cdot}{\partial t_i} = t_i(1-t_i)\frac{\partial \cdot}{\partial t_i}$ Although this is a constrained optimization problem (branch lengths cannot be negative) it is not necessary to use the log barrier method because the $1/(t_i - t_p)$ terms in the objective implicitly enforce the constraints.

3.3 Hyperparameter learning.

The DDT has two hyperparameters: the variance of the underlying Brownian motion σ^2 and the divergence function parameter c, which controls the smoothness of the data. For the full message passing framework, the overall variance σ^2 is given a Gaussian prior and variational posterior and learnt using the multiplication factor with $\alpha = 0$, corresponding to the mean field divergence measure. For the EM algorithm we use a Gamma prior and variational posterior for $1/\sigma^2$.

The message from each segment $[ab]$ to $1/\sigma^2$ is then

$$m_{[ab]\to 1/\sigma^2} = \mathrm{G}\left(\frac{D}{2}+1, \frac{b_{[pi]}}{2(t_b - t_a)}\right)$$

where $G(\alpha, \beta)$ is a Gamma distribution with shape α and rate β, and $b_{[pi]}$ is the same as for Equation 4. The smoothness c is given a Gamma prior, and sent the following VMP message from every internal node i:

$$\begin{aligned}\langle \log p(t_i, c)\rangle &= \log c + (cJ_{l(i),r(i)} - 1)\langle \log(1-t_i)\rangle \\ \Rightarrow m_{i\to c} &= \mathrm{G}\left(c; 2, -J_{l(i),r(i)}\langle \log[1-t_i]\rangle\right)\end{aligned}$$

The term $\langle \log(1-t_i)\rangle$ is deterministic for the EM algorithm and is easily approximated under the full message passing algorithm by mapping the Gaussian $q(t_i)$ to a Beta$(t_i; \alpha, \beta)$ distribution with the same mean and variance, and noting that $\langle \log(1-t_i)\rangle = \phi(\beta) - \phi(\alpha+\beta)$ where $\phi(.)$ is the digamma function.

3.4 Search over tree structures

Our resulting message passing algorithm approximates the marginal likelihood for a fixed tree structure, $p(y|\mathcal{T})p(\mathcal{T})$ (we include the factor for the probability of the tree structure itself). Ideally we would now sum the marginal likelihood over all possible tree structures $\mathcal{T}$ over N leaf nodes. Unfortunately, there are $\frac{(2N)!}{(N+1)!N!}$ such tree structures so that enumeration of all tree structures for even a modest number of leaves is not feasible. Instead we maintain a list of K-best trees (typically $K = 10$) which we find gives good empirical performance on a density estimation problem.

We search the space of tree structures by detaching and re-attaching subtrees, which may in fact be single leaves nodes. Central to the efficiency of our method is keeping the messages (and divergence times) for both the main tree and detached subtree so that small changes to the structure only require a few iterations of inference to reconverge.

We experimented with several heuristics for choosing which subtree to detach but none significantly outperformed choosing a subtree at random. However, we greatly improve upon attaching at random. We calculate the local contribution to the evidence that would be made by attaching the root of the subtree to the midpoint of each possible branch. We then run inference on the L-best attachments ($L = 3$ worked well, see Figure 5).

Sequential tree building. To build an initial tree structure we sequentially process the N leaves. We start with a single internal node with the first two leaves as children. We run inference to convergence on this tree. Given a current tree incorporating the first $n-1$ leaves, we use the local evidence calculation described above to propose L possible branches at which we could attach leaf n. We run inference to convergence on the L resulting trees and choose the one with the best evidence for the next iteration.

Tree search. Starting from a random tree or a tree built using the sequential tree building algorithm, we can use tree search to improve the list of K-best trees. We detach a subtree at random from the current best tree, and use the local evidence calculation to propose L branches at which to re-attach the detached subtree. We run message passing/EM to convergence in the resulting trees, add these to the list of trees and keep only the K best trees in terms of model evidence for the next iteration.

3.5 Predictive distribution

To calculate the predictive distribution for a specific tree we compute the distribution for a new data point conditioned on the posterior location and divergence time marginals. Firstly, we calculate the probability of diverging from each branch according to the data generating process described in Section 2. Secondly we draw several (typically three) samples of when divergence from each branch occurs. Finally we calculate the Gaussian at the leaves resulting from Brownian motion starting at the sampled divergence time and location up to to $t = 1$. This results in a predictive distribution represented as a weighted mixture of Gaussians. Finally we average the density from the K-best trees found by the algorithm.

3.6 Likelihood models

Connecting our DDT module to different likelihood models is straightforward. We demonstrate a Gaussian observation model for multivariate continuous data and a probit model for binary vectors. Both factors use $\alpha = 1$, corresponding to EP (Minka, 2001).

3.7 Computational cost

One iteration of message passing costs $O(ND)$. Message passing therefore has complexity $O(mND)$ where m is the number of iterations. m is kept small by maintaining messages when changing the structure. The E-step of EM costs $O(nND)$, where n is the number of iterations. With fixed σ and Gaussian observations the E-step is simply belief propagation so $n = 1$. Usually $n < m$ due to not updating divergence times. The order k LBFGS in the M-step costs $O(ksN)$, where s, the number of iterations, is typically small due to good initialisation. So an iteration costs $O(mND + skN)$. MCMC requires sampling the divergence times using slice sampling: a single divergence time requires an $O(ND)$ belief propagation sweep, giving total cost $O(tN^2D)$ where t is the number of samples used before sampling the structure. The different scaling with N, combined with our more directed, greedy search confers our improvement in performance.

4 Experiments

We tested our algorithms on both synthetic and real world data to assess computational and statistical performance both of variants of our algorithms and competing methods. Where computation times are given these were on a system running Windows 7 Professional with a Intel Core i7 2.67GHz quadcore processor and 4GB RAM.

Toy 2D fractal dataset. Our first experiment is on a simple two dimensional toy example with clear hierarchical (fractal) structure shown in Figure 3, with $N = 63$ datapoints. Using the message passing in EM algorithm with sequential tree building followed by 100 iterations of tree search we obtain the tree shown in Figure 3 in 7 seconds. The algorithm has recovered the underlying hierarchical structure of data apart from the occasional mistake close to the leaves where it is not clear what the optimal solution should be anyway.

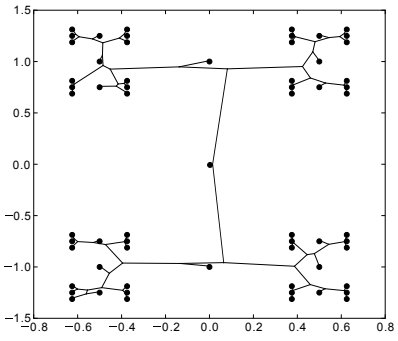

Figure 3: Toy 2D fractal dataset (N=63) showing learnt tree structure.

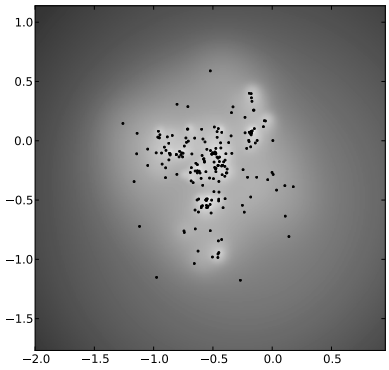

Figure 4: First two dimensions of the synthetic dataset from the prior with $D = 5, N = 200, \sigma^2 = 1, c = 1$. Lighter background denotes higher probability density.

Data from the prior ($D = 5, N = 200$). We use a dataset sampled from the prior with $\sigma^2 = 1, c = 1$, shown in Figure 4, to assess the different approaches to tree building and search discussing in Section 3.4. The results are shown in Figure 5. Eight repeats of each method were performed using different random seeds. The slowest method starts with a random tree and tries randomly re-attaching subtrees ("*search random*"). Preferentially proposing re-attaching subtrees at the best three positions significantly improves performance ("*search greedy*"). Sequential tree building is very fast (5-7 seconds), and can be followed by search where we only move leaves ("*build+search leaves*") or better, subtrees

("*build+search subtrees*"). The spread in initial log evidences for the sequential tree build methods is due to different permutations of the data used for the sequential processing. This variation suggests tree building using several random permutations of the data (potentially in parallel) and then choosing the best resulting tree.

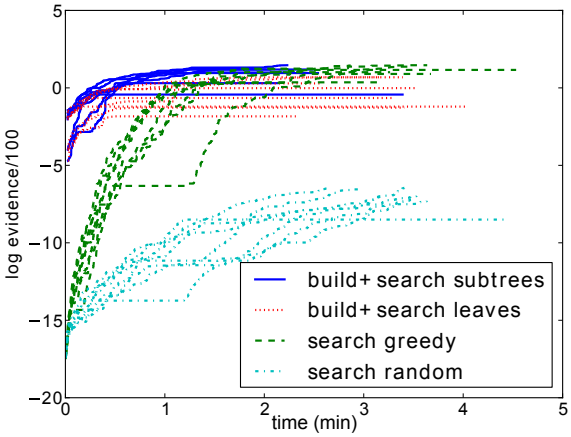

Figure 5: Performance of different tree building/search methods on the synthetic dataset.

Macaque skull measurements ($N = 200, D = 10$). We the macaque skull measurement data of Adams et al. (2008) to assess our algorithm's performance as a density model. Following Adams et al. (2008) we split the 10 dimensional data into 200 training points and 28 test points and model the three technical repeats separately. We compare to the infinite mixture of Gaussians (iMOG MCMC) and DDT MCMC methods implemented in Radford Neal's Flexible Bayesian Modeling software (see `http://www.cs.toronto.edu/~radford/`). As a baseline we use a kernel density estimate with bandwidth selected using the `npudens` R package. The results are shown in Figure 6. The EM version of our algorithm is able to find a good solution in just a few tens of seconds, but is eventually beaten on predictive performance by the MCMC solution. The full message passing solution lies between the MCMC and EM solutions in terms of speed, and only outperforms the EM solution on the first of the three repeats. The DDT based algorithms typically outperform the infinite mixture of Gaussians, with the exception of the second dataset.

Gene expression dataset ($N = 2000, D = 171$). We apply the EM algorithm with sequential tree building and 200 iterations of tree search to hierarchical clustering of the 2000 most variable genes from Yu & Landsittel (2004). We calculate predictive log likelihoods on four splits into 1800 training and 200 test

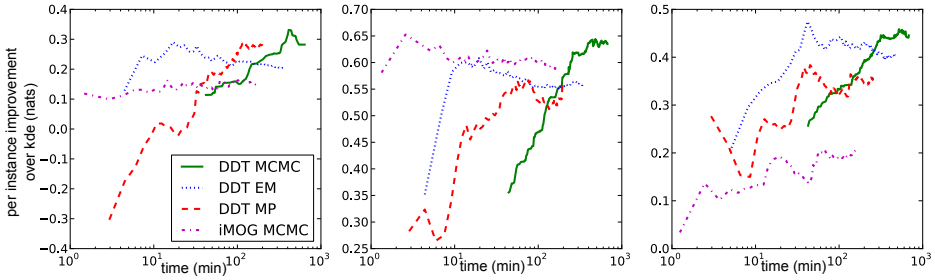

Figure 6: Per instance test set performance on the macaque skull measurement data Adams et al. (2008). The three plots arise from using the three technical replicates as separate datasets.

	iMOG	DDT EM	DDT MCMC
Score	-1.00 ± 0.04	-0.91 ± 0.02	-0.88 ± 0.03
Time	37min	48min	18hours

Table 1: Results on a gene expression dataset (Yu & Landsittel, 2004). *Score* is the per test point, per dimension log predictive likelihood. *Time* is the average computation time on the system described in Section 4.

genes. The results are shown in Table 1. The EM algorithm for the DDT has comparable statistical performance to the MCMC solution whilst being an order of magnitude faster. Both implementations significantly outperform iMOG in terms of predictive performance. DDT MCMC was run for 100 iterations, where one iteration involves sampling the position of every subtree, and the score computed averaging over the last 50 samples. Running DDT MCMC for 5 iterations takes 54min (comparable to the time for EM) and gives a score of -0.98 ± 0.04, worse than DDT EM.

Animal species. To demonstrate the use of an alternative observation model we use a probit observation model in each dimension to model 102-dimensional binary feature vectors relating to attributes (e.g. being warm-blooded, having two legs) of 33 animal species (Kemp & Tenenbaum, 2008). The tree structure we find, shown in Figure 7, is intuitive, with subtrees corresponding to land mammals, aquatic mammals, reptiles, birds, and insects (shown by colour coding).

5 Related work

We briefly discuss some relevant related work.

Kingman's coalescent (Teh et al., 2008) is similar to the Dirichlet Diffusion Tree in spirit although the generative process is defined going backwards in time

as datapoints coalesce together, rather than forward in time as for the DDT. Efficient inference for Kingman's coalescent was demonstrated in Teh & Gorur (2009). We leave investigating whether our framework could be adapted to the coalescent as future work.

In Bayesian Hierarchical Clustering (Heller & Ghahramani) the bottom-up agglomerative approach is kept but a principled probabilistic model is used to find subtrees of the hierarchy. Bayesian evidence is then used as the metric to decide which node to incorporate in the tree. The model works for continuous data but the R package only supports categorical data, whereas our quantitative assessments use continuous data. On the "Animals" example the tree obtained is broadly consistent. BHC is faster than our method, taking 3s vs 30s for DDT EM (the difference would be less for continuous data where we do not require EP). However, BHC is not a generative model and so cannot be coherently incorporated into larger models.

Roy et al. (2006) present an elegant model, but one which is only directly applicable to binary data: a sensible extension to continuous data is not obvious.

In the model of Williams (2000) each child chooses a parent in the layer above. The number of layers and nodes per layer must be pre-specified: learning this in fact gave worse results. Unlike the DDT, the model is parametric, so its complexity cannot adapt to the data.

The nested CRP Blei et al. (2010) is an alternative generative model which defines probability distributions over tree structures. One difference to the DDT though is that it does not share the property that two nodes close in the tree are correlated: rather, it models each data point as a mixture over the nodes on the path from the root to that data point. A variational inference procedure for the nCRP has recently been introduced by Wang & Blei.

6 Conclusion

Our approximate inference scheme, combining message passing and greedy tree search, is a computationally attractive alternative to MCMC for DDT models. We have demonstrated the strength of our method for modeling observed continuous and binary data at the leaves, and hope that by making code available we will encourage the community to use this elegant prior over hierarchies. In ongoing work we use the DDT to learn hierarchical structure over latent variables in models including Hidden Markov Models, specifically in part of speech tagging (Kupiec, 1992) where a hierarchy over the latent states aids interpretability, and Latent Dirichlet Allocation, where it is intuitive that topics might be hierarchically clustered (Blei et al., 2004).

References

Adams, Ryan, Murray, Iain, and MacKay, David. The Gaussian process density sampler. In *Advances in Neural Information Processing Systems*, volume 21.

MIT Press, 2008.

Aldous, D. Exchangeability and related topics. In *Ecole d'Ete de Probabilities de Saint-Flour*, volume XIII, pp. 1–198, 1985.

Beal, M. J. *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Blei, D. M., Griffiths, T. L., Jordan, M. I., and Tenenbaum, J. B. Hierarchical topic models and the nested Chinese restaurant process. *Advances in Neural Information Processing Systems*, 16:106, 2004.

Blei, David M., Griffiths, Thomas L., and Jordan, Michael I. The nested chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57, 2010.

Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2001.

Heller, K. A. and Ghahramani, Z. In *Proceedings of the 22nd International Conference on Machine learning*, pp. 304.

Kemp, Charles and Tenenbaum, Joshua B. The discovery of structural form. In *Proceedings of the National Academy of Sciences*, volume 105(31), pp. 10687–10692, 2008.

Kim, H.-C. and Ghahramani, Z. Bayesian Gaussian process classification with the EM-EP algorithm. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pp. 1948–1959, 2006.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498519, 2001.

Kupiec, Julian. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225 – 242, 1992. ISSN 0885-2308.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45:503–528, December 1989. ISSN 0025-5610.

Minka, T. P. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, 2001.

Minka, T. P. Divergence measures and message passing. *Microsoft Research, Cambridge, UK, Tech. Rep. MSR-TR-2005-173*, 2005.

Minka, T. P., Winn, J. M., Guiver, J. P., and Knowles, D. A. Infer.NET 2.4, 2010. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

Neal, R. M. Defining priors for distributions using Dirichlet diffusion trees. Technical Report 0104, Dept. of Statistics, University of Toronto, 2001.

Neal, R. M. Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statistics*, 7:619–629, 2003.

Rai, Piyush and Daumé III, Hal. The infinite hierarchical factor regression model. In *Advances in Neural Information Processing Systems*, volume 21. MIT Press, 2008.

Rasmussen, C. E. The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems*, 12:554–560, 2000.

Roy, Daniel M., Kemp, Charles, Mansinghka, Vikash K., and Tenenbaum, Joshua B. Learning annotated hierarchies from relational data. In *Advances in Neural Information Processing Systems*, volume 19, 2006.

Stern, D.H., Herbrich, R., and Graepel, T. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pp. 111120. ACM New York, NY, USA, 2009. URL `http://portal.acm.org/citation.cfm?id=1526709.1526725`.

Teh, Y. W., Daumé III, H., and Roy, D. M. Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems*, 20, 2008.

Teh, Y.W. and Gorur, D. An Efficient Sequential Monte Carlo Algorithm for Coalescent Clustering. *Advances in Neural Information Processing Systems*, 2009.

Wang, C. and Blei, D.M. Variational Inference for the Nested Chinese Restaurant Process. *cs.princeton.edu*, pp. 1–9. URL `http://www.cs.princeton.edu/~{}blei/papers/WangBlei2009d.pdf`.

Williams, C. A MCMC approach to hierarchical mixture modelling. *Advances in Neural Information Processing Systems*, 13, 2000.

Winn, J. and Bishop, C. M. Variational message passing. *Journal of Machine Learning Research*, 6(1):661, 2006.

Yu, Y. P. and Landsittel, D. et al. Gene expression alterations in prostate cancer predicting tumor aggression and preceding development of malignancy. *Journal of Clinical Oncology*, 22(14):2790–2799, Jul 2004.

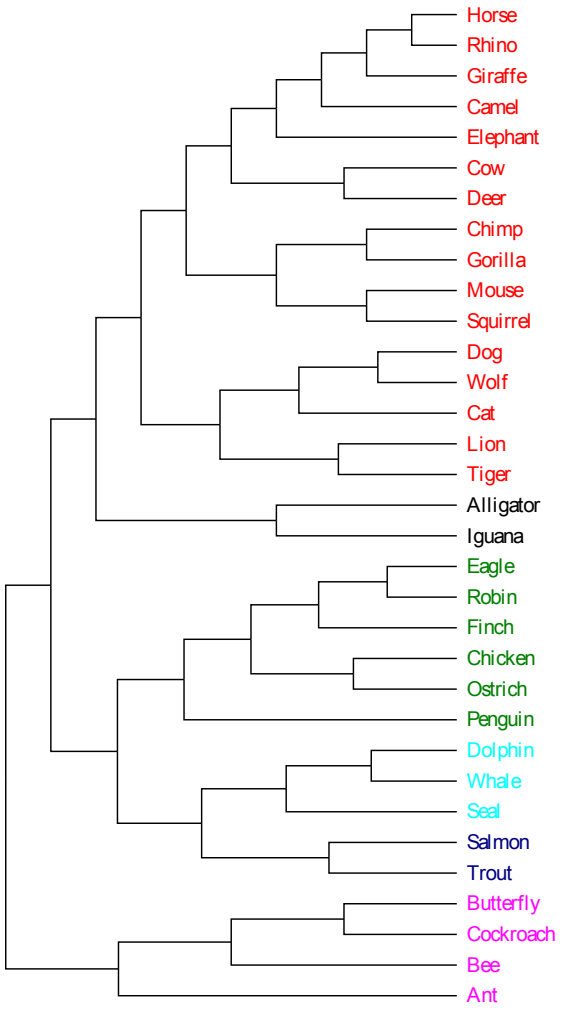

Figure 7: Tree structure learnt over animals using 102 binary features with the probit observation model.