MAP Estimation of Unobserved Variables in Conditional Gaussian Distributions

Christopher Raphael*
Department of Mathematics and Statistics
University of Massachusetts, Amherst
raphael@math.umass.edu

May 24, 2001

Abstract

We present methodology for identifying the most likely configuration of unobserved variables, given observed variables, for collections of discrete and continuous variables with conditional Gaussian (CG) distributions. The dependency structure on the joint distribution of discrete and continuous variables is represented in terms of a directed acyclic graph (DAG) in which continuous variables do not have discrete children. The joint distribution is then factored as a product of potentials, each depending only on the variables in a clique of a triangulation of the moralized DAG. A dynamic programming method is developed in which a clique potential is represented as a maximum of a finite number of Gaussian kernels. The dynamic programming progresses from the leaves of the tree toward the root recursively computing the optimal likelihood function for each subtree. The parametric representation of the optimal likelihood function, a maximum of Gaussians, is closed under the dynamic programming iteration. A "thinning" method is discussed that prevents the complexity of the representation from growing throughout the dynamic programming iterations. The implementation of the thinning algorithm is straightforward when separators in the junction tree have at most one continuous variable, however extensions are discussed for the general case. A demonstration of the method is presented on the rhythmic parsing problem: Given a sequence of times at which musical events occur, we simultaneously estimate the time varying tempo and the notated rhythm.

Keywords: conditional Gaussian distribution, MAP estimate, dynamic programming, rhythmic parsing, graphical models

1 Introduction

A graphical model represents the dependency structure of a collection of random variables through a graph. Random variables are associated with the vertices of a graph and, if a set of variables separates the graph into two "non-communicating" components, then then variables in these components are conditionally independent given the separating variables. Many situations arise in which some variables in a graphical model are observed and one seeks the most likely configuration of unobserved variables — the *maximum a posteriori* (MAP) estimate. A well-known example is the hidden Markov model (HMM) approach to speech recognition in which the estimated sequence of words follows directly from the most likely sequence of hidden states [1] given the observed acoustical signal. In the case of HMMs, the identification of the MAP estimate relies on the linear structure of the graph, however generalizations are possible. In particular, methodology for identifying MAP estimates for graphical models with arbitrary graph structures is well-known for the case of fully discrete networks [2], [3].

Applications modeled by *mixed* collections of discrete and continuous variables are ubiquitous, however the inference methodology for such mixed models is less well developed. We consider a particular class of mixed models, conditional Gaussian (CG) distributions, in which, given each configuration of discrete variables, the

*This work is supported by NSF grant IIS-9987898.

conditional distribution on the continuous variables is multivariate Gaussian [4], [5]. CG distributions and the corresponding methodology involved in propagating evidence and computing local marginal distributions has been developed by Lauritzen [6], and Lauritzen and Jensen [7]. While Lauritzen and Jensen give a general method for computing local marginals, the approach is only feasible when the interface between continuous and discrete variables is small due to the reliance on a graph with a "strong root." In this paper we show how to compute MAP estimates in networks representing arbitrary CG distributions. Since our methodology does not require a graph with a strong root, it is feasible in cases in which one could not realistically compute marginal distributions.

The outline of our paper is as follows. Section 2 gives our dynamic programming algorithm for constructing MAP estimates for networks representing CG distributions. Section 3 discusses an important implementational issue surrounding an aspect of the algorithm — the "thinning" procedure. Exact implementation of this procedure is straightforward when separators of the junction tree have only one continuous variable and we present an algorithm for doing so. Section 3 also discusses more general approaches to thinning. Section 4 demonstrates an application of our methods to the *rhythmic parsing* problem encountered in music information retrieval. Section 5 contains a discussion of our MAP approach and its relation to Lauritzen's method. Finally, Section 6 presents some easily derived facts about Gaussian kernels and is intended as reference for the other sections.

2 Computing MAP Estimates for CG Distributions

2.1 CG Distributions

Let X be a random vector whose components are indexed by a set V. We write X_A for the subset of components indexed by $A \subseteq V$ so $X = X_V$. Suppose the probability distribution for X can be represented in terms of a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$. More precisely, we assume that the probability density function for X, $f(x)$, can be factored as a product of conditional densities

$$f(x) = \prod_{v \in V} f_v(x_v | x_{\mathrm{pa}(v)}) \tag{1}$$

where $\mathrm{pa}(x_v)$ are the parents of v in $\mathcal{G}$.

We assume that X has a *conditional Gaussian* (CG) distribution [4], [5]. That is, V is partitioned by $V = \Delta \cup \Gamma$ where Δ and Γ are disjoint sets indexing the discrete and continuous variables of X. We will use the notation $\Delta(C) = \Delta \cap C$ and $\Gamma(C) = \Gamma \cap C$. For a CG distribution, every discrete variable, X_v, $v \in \Delta$ can take on a finite number of values while the conditional distribution of X_Γ given X_Δ is multivariate normal:

$$\mathcal{L}(X_\Gamma | X_\Delta = x_\Delta) = N(\mu(x_\Delta), \Sigma(x_\Delta))$$

In terms of the DAG representation, for X to have a CG distribution is it sufficient that for $v \in \Delta$, $\mathrm{pa}(v) \cap \Gamma = \emptyset$ (discrete nodes do not have continuous parents), and for $v \in \Gamma$,

$$f_v(x_v | x_{\mathrm{pa}(v)}) = N(x_v; \alpha^t(x_D) x_G + \beta(x_D), \sigma^2(x_D))$$

where $D = \Delta(\mathrm{pa}(v))$, $G = \Gamma(\mathrm{pa}(v))$ and $N(x; \mu, \sigma^2)$ is the univariate normal density function. Such conditional distributions are deemed *CG regressions* by Lauritzen and Wermuth [5],[7].

Suppose that V is also partitioned into two sets, O and U, indexing the observed and unobserved variables. We can represent the conditional distribution of X given the observed variables in terms of an undirected graph, $\bar{\mathcal{G}}$, as follows. We construct $\bar{\mathcal{G}}$ in the following three-stage procedure:

Moralize We moralize $\mathcal{G}$ by dropping the directions of the arcs and connecting the parents of each node. Each factor of Eqn. 1 can then be represented as a function of the variables of a clique in the moralized graph.

Restrict We restrict the graph to U by eliminating all nodes in O and all arcs that are incident with O. For $v \in V$ let $\bar{f}_v$ be f_v from Eqn. 1 with the variables in O held fixed. Thus $\bar{f}_v$ does not depend on any variables in O and is a function of the variables of a clique in the restricted graph.

Triangulate Triangulate the moralized and restricted graph resulting in $\bar{\mathcal{G}}$. Thus, for each $v \in V$, $\bar{f}_v$ is a function of the variables of a clique in $\bar{\mathcal{G}}$, $C(v)$.

An example of this procedure is given in Figure 2.

Suppose that $\mathcal{T}$ is a junction tree for $\bar{\mathcal{G}}$ with cliques $\mathcal{C}$ and separators $\mathcal{S}$. Given $X_O = x_O$, we can then represent the conditional distribution on the unobserved variables, $\bar{f}(x_U)$, as

$$\begin{aligned} \bar{f}(x_U) &\propto f(x_U, x_O) \\ &= \prod_{v \in V} \bar{f}_v \\ &= \prod_{C \in \mathcal{C}} \prod_{C(v)=C} \bar{f}_v \qquad (2) \\ &= \prod_{C \in \mathcal{C}} \phi_C(x_C) \qquad (3) \end{aligned}$$

where $\phi_C(x_C) = \prod_{C(v)=C} \bar{f}_v$. Thus the conditional density, $\bar{f}(x_U)$, can be represented as a product of potentials, each depending only on the variables of a clique in a triangulated graph.

2.2 Dynamic Programming on Junction Trees

Let x be a vector whose components are indexed by a finite set V. The components of x can be discrete variables, continuous variables, or any mixture of the two. We continue to write x_C for the components of x indexed by $C \subseteq V$.

We now consider the problem of maximizing a function, $f(x)$, of the form

$$df(x) = \prod_{C \in \mathcal{C}} \phi_C(x_C) \qquad (4)$$

where $\mathcal{C}$ are the cliques of a junction tree $\mathcal{T}$. In particular, the cliques have the property that, for $C, C_1, C_2 \in \mathcal{C}$, if C is on the unique path connecting C_1 and C_2 in $\mathcal{T}$, then $C_1 \cap C_2 \subseteq C$.

We arbitrarily choose an element $C_r \in \mathcal{C}$ to be the "root" of the junction tree. We do not require that C_r be a "strong root" [3] as does the propagation scheme of Lauritzen [6]. The implications of this issue are discussed in Section 5 Choosing a root imposes a partial ordering on $\mathcal{C}$; we will use the notation $\tilde{C} < C$ to mean that C lies on the unique path between $\tilde{C}$ and C_r in $\mathcal{T}$ (our trees grow downward from the root), $\tilde{C} \leq C$ to mean $\tilde{C} = C$ or $\tilde{C} < C$, and $C \overset{S}{\to} \tilde{C}$ to mean that C and $\tilde{C}$ are neighboring cliques of $\mathcal{T}$ separated by $S = C \cap \tilde{C}$ with $\tilde{C} < C$.

The optimal value, $\max_x f(x)$, can then be found by dynamic programming as follows. Define

$$f_C(x) = \prod_{\tilde{C} \leq C} \phi_{\tilde{C}}(x_{\tilde{C}})$$

Note that $f_C(x)$ only depends on the variables of x contained in cliques, $\tilde{C}$, with $\tilde{C} \leq C$, however we do not make this explicit in the notation. Define

$$\begin{aligned} H_C(x_C) &= \max_{x_{V \setminus C}} f_C(x) \qquad (5) \\ &= \max_{x_{V \setminus C}} f_C(x_C, x_{V \setminus C}) \end{aligned}$$

and observe that $H_C(x_C)$ can be computed recursively by

$$\begin{aligned} H_C(x_C) &= \max_{x_{V \setminus C}} \prod_{\tilde{C} \leq C} \phi_{\tilde{C}}(x_{\tilde{C}}) \\ &= \phi_C(x_C) \max_{x_{V \setminus C}} \prod_{C \overset{S}{\to} \tilde{C}} \prod_{\bar{C} \leq \tilde{C}} \phi_{\bar{C}}(x_{\bar{C}}) \end{aligned}$$

$$= \phi_C(x_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{V \setminus C}} \prod_{\bar{C} \leq \tilde{C}} \phi_{\bar{C}}(x_{\bar{C}}) \quad (6)$$

$$= \phi_C(x_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{V \setminus S}} \prod_{\bar{C} \leq \tilde{C}} \phi_{\bar{C}}(x_{\bar{C}})$$

$$= \phi_C(x_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{\tilde{C} \setminus S}} \max_{x_{V \setminus \tilde{C}}} \prod_{\bar{C} \leq \tilde{C}} \phi_{\bar{C}}(x_{\bar{C}})$$

$$= \phi_C(x_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{\tilde{C} \setminus S}} H_{\tilde{C}}(x_{\tilde{C}}) \quad (7)$$

where the exchange of maximization and product in Eqn. 6 is justified because if $\tilde{C}_1$ and $\tilde{C}_2$ are two children of C, i. e. $C \xrightarrow{S_j} \tilde{C}_j$ for $j = 1, 2$, then, because $\mathcal{C}$ are the cliques of a junction tree, the variables common to $\prod_{\bar{C} \leq \tilde{C}_j} \phi_{\bar{C}}(x_{\bar{C}})$, $j = 1, 2$, must also be in C, and hence held fixed in the maximization.

When H_C has been computed for all $C \in \mathcal{C}$ we have

$$\begin{aligned} \max_{x_{C_r}} H_{C_r}(x_{C_r}) &= \max_{x_{C_r}} \max_{x_{V \setminus C_r}} f_{C_r}(x) \\ &= \max_{x} f(x) \end{aligned}$$

It is then a simple matter to recover a maximizing configuration of $f(x)$. Define $\hat{x}_C$ for each $C \in \mathcal{C}$ as follows. For the root clique we let

$$\hat{x}_{C_r} = \arg\max_{x_{C_r}} H(x_{C_r}) \quad (8)$$

where we take *any* maximizer when the maximizer is not unique. Otherwise, if $\hat{x}_C$ is defined and $C \xrightarrow{S} \tilde{C}$ then $S \subseteq C$ so $\hat{x}_S$ is defined. We then let $\hat{x}_{\tilde{C}} = (\hat{x}_S, \hat{x}_{\tilde{C} \setminus S})$ where

$$\hat{x}_{\tilde{C} \setminus S} = \arg\max_{x_{\tilde{C} \setminus S}} H_{\tilde{C}}(\hat{x}_S, x_{\tilde{C} \setminus S}) \quad (9)$$

If we then consider Eqn. 7 for $\hat{x}_C$ we have

$$\begin{aligned} H_C(\hat{x}_C) &= \phi_C(\hat{x}_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{\tilde{C} \setminus S}} H_{\tilde{C}}(\hat{x}_S, x_{\tilde{C} \setminus S}) \\ &= \phi_C(\hat{x}_C) \prod_{C \xrightarrow{S} \tilde{C}} H_{\tilde{C}}(\hat{x}_{\tilde{C}}) \end{aligned}$$

Iterative application of this result gives $H_{C_r}(\hat{x}_{C_r}) = \prod_{C \in \mathcal{C}} \phi_C(\hat{x}_C)$ and from this it follows that

$$\begin{aligned} f(\hat{x}) &= \prod_{C \in \mathcal{C}} \phi_C(\hat{x}_C) \\ &= H_{C_r}(\hat{x}_{C_r}) \\ &= \max_{x} f(x) \end{aligned}$$

2.3 Maximizing the Likelihood Function

We now consider the problem of finding a MAP estimate of the unobserved variables, X_U, given the observation $X_O = x_O$ when X has a CG distribution. That is, we seek

$$\hat{x}_U = \arg\max_{x_U} \bar{f}(x_U)$$

where $\bar{f}(x_U)$ is the conditional density at configuration x_U. Note from Eqn. 3 that $\bar{f}(x_U)$ can be expressed as a product of potentials where each potential depends only on the variables of a clique in a triangulated graph, as in Eqn. 4. In Section 2.2 we showed that such a function can, in principle, be maximized using dynamic

programming. The implementation of the basic DP iteration in Eqn. 7 is completely straightforward in the case of functions of discrete variables. In the CG case we treat here, some of the variables are continuous, hence our representation of the objective function $\bar{f}$ must be parametric. In such a case the computation of the DP iteration must be carried out in terms of the parameters of the representation, and hence is not as simple as in the discrete case. In this section we show that, for CG distributions, the DP iteration can be carried out exactly. In particular, we show that the functions H_C, $C \in \mathcal{C}$, defined in Eqn. 5 will have a specific parametric representation which is *closed* under the DP iteration. Thus it is possible to carry out the dynamic programming computations in terms of the parameters of the representation.

Define the n-dimensional *Gaussian kernel*

$$K(x;\theta) = K(x;h,m,Q) = he^{-\frac{1}{2}(x-m)^t Q(x-m)} \tag{10}$$

where x is an n-vector, h is a nonnegative constant, m is an n-vector, and Q is an $n \times n$ nonnegative definite matrix. We write $\theta = (h(\theta), m(\theta), Q(\theta))$ to represent the components of θ. It is possible to perform a number of operations on Gaussian kernels such as multiplication of two kernels, maximizing over a subset of variables, and representing conditional Gaussian distributions by performing transformations of the parameters involved. Section 6 gives an account of some easily derived results involving Gaussian kernels. We refer to this section occasionally throughout the current section.

Consider the following class of functions for $C \in \mathcal{C}$:

$$\mathcal{M}_C = \begin{cases} \{H(x_C) : H(x_C) \geq 0 \; \forall \; x_C\} & \Gamma(C) = \emptyset \\ \{H(x_C) : H(x_C) = \max_{\theta \in \Theta(x_{\Delta(C)})} K(x_{\Gamma(C)};\theta)\} & \text{otherwise} \end{cases} \tag{11}$$

Thus a function in $\mathcal{M}_C$ is just the usual potential function when C only contains discrete variables. However, when C contains continuous variables, every configuration of discrete variables gives rise to a separate maximum of Gaussian kernels. In interpreting the above maximum when $\Delta(C) = \emptyset$, there is only one possible configuration of $x_{\Delta(C)}$ — the "null" configuration, however, $\Theta(x_{\Delta(C)})$ might still contain several parameter values.

Now observe that each ϕ_C in Eqn. 3 can be represented as a function in $\mathcal{M}_C$. This fact is obvious in the cases where C contains only discrete variables. Otherwise note that f_v in Eqn. 1 can be written as

$$f_v(x_v|x_{\mathrm{pa}(v)}) = K(x_{\Gamma(\mathrm{fa}(v))};\theta(x_{\Delta(\mathrm{fa}(v))}))$$

where the *family* of v is defined by $\mathrm{fa}(v) = v \cup \mathrm{pa}(v)$, and the conditional distribution is computed using Eqn. 24 for each configuration of discrete parents. If there are no discrete parents, then $x_{\Delta(\mathrm{fa}(v))}$ can only be the null configuration and $\theta = \theta(x_{\Delta(\mathrm{fa}(v))})$ is the single parameter value describing the conditional distribution given in Eqn. 24. Then $\bar{f}_v$, which is f_v with the observed variables held fixed, (and hence eliminated), can be computed as

$$\bar{f}_v(x_{\mathrm{fa}(v)\cap U}) = K(x_{\Gamma(\mathrm{fa}(v)\cap U)};\theta(x_{\Delta(\mathrm{fa}(v)\cap U)}))$$

using Eqn. 22. Next $\bar{f}_v$ is extended to be a function of $x_{C(v)}$, $\tilde{f}_v$, using Eqn. 23 and extending to any discrete variables in $C \setminus (\mathrm{fa}(v) \cap U)$ in the obvious way. Finally $\phi_C = \prod_{C(v)=C} \bar{f}_v = \prod_{C(v)=C} \tilde{f}_v$ is computed by repeated application of Eqn. 18 giving a potential of the form

$$\phi_C(x_C) = K(x_{\Gamma(C)};\theta(x_{\Delta(C)}))$$

The preceding argument shows that for terminal cliques in the junction tree, C_t, we have from Eqn. 5 $H_{C_t} = \phi_{C_t} \in \mathcal{M}_{C_t}$. We now show that $H_C \in \mathcal{M}_C$ for all C by observing that the operations of "maxing out," extension, and product encountered in Eqn. 7 are all expressible in terms of transformations of the parameters of our representation.

Maxing Out Suppose $H(x_C) \in \mathcal{M}_C$ with $S \subset C$ and let $R = C \setminus S$. We will show that $\max_{x_R} H(x_C) = \max_{x_R} H(x_S, x_R) \in \mathcal{M}_S$. There are three cases to consider

1. $\Gamma(C) = \emptyset$
2. $\Gamma(C) \neq \emptyset$ and $\Gamma(S) \neq \emptyset$

3. $\Gamma(C) \neq \emptyset$ and $\Gamma(S) = \emptyset$

If $\Gamma(C) = \emptyset$, $\max_{x_R} H(x_C)$ is a discrete potential and clearly in $\mathcal{M}_S$.
If $\Gamma(C) \neq \emptyset$ and $\Gamma(S) \neq \emptyset$,

$$\begin{aligned}
\max_{x_R} H(x_C) &= \max_{x_R} \max_{\theta \in \Theta(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta) \\
&= \max_{x_{\Delta(R)}} \max_{\theta \in \Theta(x_{\Delta(C)})} \max_{x_{\Gamma(R)}} K(x_{\Gamma(S)}, x_{\Gamma(R)}; \theta) \\
&= \max_{x_{\Delta(R)}} \max_{\theta \in \Theta(x_{\Delta(S)}, x_{\Delta(R)})} K(x_{\Gamma(S)}; M_S(\theta)) \qquad (12) \\
&= \max_{\theta \in \tilde{\Theta}(x_{\Delta(S)})} K(x_{\Gamma(S)}; \theta) \qquad (13) \\
&= \max_{\theta \in \Theta(x_{\Delta(S)})} K(x_{\Gamma(S)}; \theta) \in \mathcal{M}_S \qquad (14)
\end{aligned}$$

where in Eqn. 12, $M_S(\theta)$ is the result of maxing out the variables of $\Gamma(R)$ as in Eqn. 21, ($M_S(\theta) = \theta$ when $\Gamma(R) = \emptyset$), and

$$\tilde{\Theta}(x_{\Delta(S)}) = \bigcup_{x_{\Delta(R)}, \theta \in \Theta(x_{\Delta(S)}, x_{\Delta(R)})} M_S(\theta)$$

and

$$\Theta(x_{\Delta(S)}) = \text{Thin}(\tilde{\Theta}(x_{\Delta(S)}))$$

where $\text{Thin}(\Theta)$ is the smallest subset of Θ such that

$$\max_{\theta \in \text{Thin}(\Theta)} K(x; \theta) = \max_{\theta \in \Theta} K(x; \theta) \qquad (15)$$

If $\Gamma(C) \neq \emptyset$ and $\Gamma(S) = \emptyset$,

$$\begin{aligned}
\max_{x_R} H(x_C) &= \max_{x_R} \max_{\theta \in \Theta(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta) \\
&= \max_{x_{\Delta(R)}} \max_{\theta \in \Theta(x_{\Delta(C)})} \max_{x_{\Gamma(R)}} K(x_{\Gamma(R)}; \theta) \\
&= \max_{x_{\Delta(R)}} \max_{\theta \in \Theta(x_S, x_{\Delta(R)})} h(\theta) \in \mathcal{M}_S
\end{aligned}$$

Extension In order to compute the product in Eqn. 7 we must first extend the factors to be functions of the variables of C. Suppose $S \subseteq C$. There are three cases to consider:

1. $\Gamma(C) = \emptyset$
2. $\Gamma(S) = \emptyset$ and $\Gamma(C) \neq \emptyset$
3. $\Gamma(S) \neq \emptyset$

When $\Gamma(C) = \emptyset$ this is simply the familiar extension of discrete potentials.
When $\Gamma(S) = \emptyset$ and $\Gamma(C) \neq \emptyset$ we extend $H_S(x_S)$ to $H_C(x_C)$ by letting $H_C(x_C) = K(x_{\Gamma(C)}; \theta(x_{\Delta(C)})$ where

$$\begin{aligned}
h(\theta(x_{\Delta(C)})) &= H_S(x_S) \\
m(\theta(x_{\Delta(C)})) &= 0 \\
Q(\theta(x_{\Delta(C)})) &= 0
\end{aligned}$$

When $\Gamma(S) \neq \emptyset$ we have $H_S(x_S) = \max_{\theta \in \Theta(x_{\Delta(S)})} K(x_{\Gamma(S)}; \theta)$. Then the extension $H_C(x_C)$ is given by

$$\begin{aligned}
H_C(x_C) &= \max_{\theta \in \Theta(x_{\Delta(S)})} K(x_{\Gamma(C)}; E(\theta)) \\
&= \max_{\theta \in \Theta(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta)
\end{aligned}$$

where $E(\theta)$ is the transformation defined in Eqn. 23 and for any configuration $x_{\Delta(C)}$,

$$\Theta(x_{\Delta(C)}) = \{E(\theta) : \theta \in \Theta(x_{\Delta(S)})\}$$

Multiplication Suppose $H = \prod_{i=1}^{N} H_i$ with $H_i \in \mathcal{M}_C$, $i = 1, \ldots, N$. If $\Gamma(C) = \emptyset$ then clearly $H \in \mathcal{M}_C$. Otherwise

$$\begin{aligned} H(x_C) &= \prod_{i=1}^{N} \max_{\theta_i \in \Theta_i(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta_i) \\ &= \max_{(\theta_1,\ldots,\theta_N)\in\Theta^*} \prod_{i=1}^{N} K(x_{\Gamma(C)}; \theta_i) \\ &= \max_{(\theta_1,\ldots,\theta_N)\in\Theta^*} K(x_{\Gamma(C)}; P(\theta_1, \ldots, \theta_N)) \\ &= \max_{\theta\in\Theta(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta) \in \mathcal{M}_C \end{aligned}$$

where

$$\Theta^* = \Theta_1(x_{\Delta(C)}) \times \ldots \times \Theta_N(x_{\Delta(C)})$$

$P(\theta_1, \ldots, \theta_N)$ is defined using Eqn. 18, and

$$\Theta(x_{\Delta(C)}) = \text{Thin}(\{P(\theta_1, \ldots, \theta_N) : (\theta_1, \ldots, \theta_N) \in \Theta^*\})$$

2.4 Recovering an Optimal Configuration

Once we have computed $H_C(x_C)$ for all $C \in \mathcal{C}$ we can recover an optimal configuration $\hat{x}$ using Eqns. 8 and 9. Here we show how to perform these computations for the particular parametric form of H_C given in Eqn. 11 resulting from CG distributions.

For the root configuration $\hat{x}_{C_r}$ there are two cases to consider:

1. $\Gamma(C_r) = \emptyset$
2. $\Gamma(C_r) \neq \emptyset$

When $\Gamma(C_r) = \emptyset$ we have $\hat{x}_{C_r} = \arg\max_{x_{C_r}} H_{C_r}(x_{C_r})$.
When $\Delta(C_r) = \emptyset$ then

$$\begin{aligned} \hat{x}_{C_r} &= \arg\max_{x_{C_r}} H_{C_r}(x_{C_r}) \\ &= \arg\max_{x_{C_r}} K(x_{C_r}; \theta) \\ &= m(\theta) \end{aligned}$$

When $\Gamma(C_r) \neq \emptyset$ then

$$\begin{aligned} \hat{x}_{C_r} &= \arg\max_{x_{C_r}} H_{C_r}(x_{C_r}) \\ &= \arg\max_{x_{C_r}} \max_{\theta\in\Theta(x_{\Delta(C_r)})} K(x_{\Gamma(C_r)}; \theta) \end{aligned}$$

Since $\max_x K(x; \theta) = h(\theta)$ and $\arg\max_x K(x; \theta) = m(\theta)$ it follows that $\hat{x}_{C_r}$ can then be computed by

$$\begin{aligned} \hat{x}_{\Delta(C_r)} &= \arg\max_{x_{\Delta(C_r)}} \max_{\theta\in\Theta(x_{\Delta(C_r)})} h(\theta) \\ \hat{\theta} &= \max_{\theta\in\Theta(\hat{x}_{\Delta(C_r)})} h(\theta) \\ \hat{x}_{\Gamma(C_r)} &= m(\hat{\theta}) \end{aligned}$$

If $\Delta C_r = \emptyset$ we needn't bother with the first equation computing $\hat{x}_{\Delta C_r}$

Similarly, once we have computed $\hat{x}_C$ then, if $C \xrightarrow{S} \tilde{C}$, we can compute $\hat{x}_R = \arg\max_{x_R} H_{\tilde{C}}(\hat{x}_S, x_R)$ where $R = \tilde{C} \setminus S$. There are three cases to consider:

1. $\Gamma(R) \neq \emptyset$
2. $\Gamma(R) = \emptyset$, $\Gamma(S) \neq \emptyset$
3. $\Gamma(R) = \emptyset$, $\Gamma(S) = \emptyset$

When $\Gamma(R) \neq \emptyset$

$$\begin{aligned}\hat{x}_R &= \arg\max_{x_R} H_{\tilde{C}}(\hat{x}_S, x_R) \\ &= \arg\max_{x_R} \max_{\theta \in \Theta(\hat{x}_{\Delta(S)}, x_{\Delta(R)})} K(\hat{x}_{\Gamma(S)}, x_{\Gamma(R)}; \theta) \\ &= \arg\max_{x_R} \max_{\theta \in \Theta(\hat{x}_{\Delta(S)}, x_{\Delta(R)})} K(x_{\Gamma(R)}; F(\theta))\end{aligned}$$

where $F(\theta)$ is found by treating $\hat{x}_{\Gamma(S)}$ as fixed and using Eqn. 22. In the event that $\Gamma(S) = \emptyset$ in the above we take $F(\theta) = \theta$. We then solve for x_R through

$$\begin{aligned}\hat{x}_{\Delta(R)} &= \arg\max_{x_{\Delta(R)}} \max_{\theta \in \Theta(\hat{x}_{\Delta(S)}, x_{\Delta(R)})} h(F(\theta)) \\ \hat{\theta} &= \arg\max_{\theta \in \Theta(\hat{x}_{\Delta(S)}, \hat{x}_{\Delta(R)})} h(F(\theta)) \\ \hat{x}_{\Gamma(R)} &= m(F(\hat{\theta}))\end{aligned}$$

Again, if $\Delta C_r = \emptyset$ we needn't bother with the first equation computing $\hat{x}_{\Delta C_r}$
When $\Gamma(R) = \emptyset$, $\Gamma(S) \neq \emptyset$

$$\begin{aligned}\hat{x}_R &= \arg\max_{x_{\Delta(R)}} H_{\tilde{C}}(\hat{x}_S, x_R) \\ &= \arg\max_{x_R} (\max_{\theta \in \Theta(\hat{x}_{\Delta(S)}, x_{\Delta(R)})} K(\hat{x}_{\Gamma(S)}; \theta))\end{aligned}$$

which is straightforward to compute since $\hat{x}_{\Gamma(S)}$ is fixed.
When $\Gamma(R) = \emptyset$, $\Gamma(S) = \emptyset$ there are no continuous variables involved and

$$\hat{x}_{\Delta(R)} = \arg\max_{x_{\Delta(R)}} H_{\tilde{C}}(\hat{x}_S, x_R)$$

3 Thinning

From a computational standpoint the optimization process of Section 2.3 is, for the most part, quite straightforward; the one exception is the "thinning" operation. Recall that $\text{Thin}(\Theta)$ is the smallest subset of Θ for which Eqn. 15 holds. While the thinning operation is well-defined, its computation can be problematic in some cases.

3.1 One-Dimensional Thinning

Thinning a set Θ is particularly simple when Θ is composed of parameters for one-dimensional Gaussian kernels. In such a case, we can incrementally construct $\text{Thin}(\Theta)$ by adding kernels, one at a time, while maintaining a partition of the real line and associating the maximizing kernel with each interval in the partition. In this algorithm, kernels that do not attain the maximum on any of the intervals are weeded out. This is made precise in the following.

One-Dimensional Thinning Algorithm Suppose $\Theta = \{\theta^1, \ldots, \theta^I\}$. Define

$$\hat{\theta}^i(t) = \arg\max_{\theta \in \{\theta^1, \ldots, \theta^i\}} K(t; \theta)$$

for $i = 1 \ldots I$ and note that $\hat{\theta}^i(t)$ is piecewise constant and, hence, can be written as

$$\hat{\theta}^i(t) = \sum_{k=1}^{N(i)} \theta_k^i 1_{(x_k^i, x_{k+1}^i)}(t)$$

where $-\infty = x_1^i < x_2^i < \ldots < x_{N(i)}^i < x_{N(i)+1}^i = \infty$ and $\theta_k^i \in \{\theta^1, \ldots, \theta^i\}$. We need not be concerned with the definition of $\hat{\theta}^i(t)$ at points, $t = x_k^i$, where the maximizer is not unique. Clearly then $\mathrm{Thin}(\Theta) = \cup_{k=1}^{N(I)} \theta_k^I$.

Note that $\hat{\theta}^i(t)$ can be computed iteratively by letting $\hat{\theta}^1(t) \equiv \theta^1$ and noting

$$\hat{\theta}^i(t) = \arg \max_{\theta \in \{\hat{\theta}^{i-1}(t), \theta^i\}} K(t; \theta) \tag{16}$$

$\hat{\theta}^i(t)$ can then be computed on each interval $(x_k^{i-1}, x_{k+1}^{i-1})$ where it must be that $\hat{\theta}^{i-1}(t) = \theta_k^{i-1}$. To do this we simply find all solutions, t, to the quadratic equation

$$\log K(t; \theta^i) - \log K(t; \theta_k^{i-1}) = 0 \tag{17}$$

which lie in $(x_k^{i-1}, x_{k+1}^{i-1})$. These points partition the interval $(x_k^{i-1}, x_{k+1}^{i-1})$ into subintervals where $\hat{\theta}^i(t)$ must be constant so we need only identify $\hat{\theta}^i(t)$ through Eqn. 16 at any interior point of these subintervals. Having done this for each interval $(x_k^{i-1}, x_{k+1}^{i-1})$ we may find that $\hat{\theta}^i(t)$ is constant over neighboring subintervals. In such a case, the neighbors are simply merged together to form a more compact representation of $\hat{\theta}^i(t)$ $\square$

An interesting variation on the thinning algorithm is as follows. Suppose we define $\mathrm{Thin}_D(\Theta)$ as the smallest subset of Θ such that Eqn. 15 holds for all x in some interval D. We call this the *constrained* thinning procedure. Clearly $\mathrm{Thin}_D(\Theta)$ can be computed using the one-dimensional thinning algorithm restricted to the interval D and the result satisfies $\mathrm{Thin}_D(\Theta) \subseteq \mathrm{Thin}(\Theta)$. Suppose that we carry through our dynamic programming algorithm with Thin_D instead of Thin and the resulting optimal configuration is $\hat{x}$. Now consider a *constrained* MAP estimate defined by

$$\hat{x}_D = \arg \max_{x : x_\Gamma \in D^{|\Gamma|}} f(x)$$

One can show that if $\hat{x}_\Gamma \in D^{|\Gamma|}$, then $\hat{x}_D = \hat{x}$. Thus if we carry though the dynamic programming procedure with using Thin_D and our optimal configuration happens to satisfy the constraints, then we have found the constrained optimum. On the other hand, when the continuous components of $\hat{x}$ are all reasonably close to D, a continuity argument suggests that $f(\hat{x})$ is close to $f(\hat{x}_D)$, thus giving an approximate constrained MAP estimate. Using constrained thinning can be appropriate from a modeling point of view and can lead to a dramatic decrease in the computational cost of our algorithm as well by decreasing the size of our representation of potential functions.

3.2 Thinning Beyond One Dimension

The one-dimensional thinning algorithm does not generalize to higher dimensions since, when Θ contains a collection of n-dimensional parameters, the partition of $\Re^n$ generated by the function

$$\hat{\theta}(x) = \arg \max_{\theta \in \Theta} K(x; \theta)$$

is difficult to represent and modify as more kernels are added to Θ. However, there are a number of different approaches to thinning in higher dimensions which we discuss here.

One possible key to higher dimensional thinning is the following. If θ_1 and θ_2 are parameters for two-dimensional Gaussian kernels, then it is a simple matter to determine if $K(x; \theta_1) > K(x; \theta_2)$ for all x in a finite two-dimensional rectangle D. We need only check to see if the quadratic function $A(x) = \log(K(x; \theta_1)) - \log(K(x; \theta_2))$ is strictly positive on D, or equivalently, if A is positive both on the boundary of D and at its global minimum, if one exists and is in the interior of D. It is completely straightforward to check if the quadratic function A has a global minimum insided D. To check if A is positive on the boundary of D we restrict A to the one-dimensional edges of the rectangle and use the quadratic formula.

Similarly, for three-dimensional parameters θ_1 and θ_2 and a three-dimensional finite rectangle D, $K(x; \theta_1) > K(x; \theta_2)$ for all $x \in D$ if and only if $A(x) = \log(K(x; \theta_1)) - \log(K(x; \theta_2))$ is positive on the boundary of D and also at its global minimum, if one exists and is inside of D. It is simple to check the latter condition. To

verify that A is positive on the boundary of D we simply apply the previous argument to A restricted to the two-dimensional faces of D. This argument can be extended recursively to determine if $K(x;\theta_1) > K(x;\theta_2)$ on D for arbitrarily dimensioned parameters and rectangles.

This idea could form the basis of a thinning algorithm for arbitrary dimensions. Suppose D is a finite rectangular subset of $\Re^n$, Θ is a finite collection of n-dimensional parameters, and we wish to perform thinning on D. As mentioned at the end of Section 3.1, this desire might be motivated by a desire to construct a constrained MAP estimate or by an interest in controlling the size of our potential representations for computational purposes. Consider an algorithm that recursively subdivides D into smaller and smaller rectangles and let $D_1^k, \ldots, D_{N(k)}^k$ be the partition obtained after the kth iteration of the algorithm. Define Θ_j^k, $j = 1, \ldots, N(k)$ to be the collection of parameter values in Θ whose kernels are not "dominated" on D_j^k by some other kernel with parameter in Θ. That is $\Theta_j^k = \Theta \setminus \Phi_j^k$ where

$$\Phi_j^k = \{\theta \in \Theta : \text{ there exists } \theta' \in \Theta \text{ such that } K(x;\theta) < K(x;\theta') \text{ for all } x \in D_j^k\}$$

If $|\Theta_j^k| = 1$ for some j then clearly $\Theta_j^k \subseteq \mathrm{Thin}(\Theta)$. In fact, if $|\Theta_j^k| = 2$ for some j then also $\Theta_j^k \subseteq \mathrm{Thin}(\Theta)$, since the two kernels dominate all other kernels on Θ_j^k and neither kernel dominates the other. Thus if at some iteration k^*

$$\bigcup_j \Theta_j^{k^*} = \bigcup_{j:|\Theta_j^{k^*}|\leq 2} \Theta_j^{k^*}$$

then $\mathrm{Thin}(\Theta) = \cup_j \Theta_j^{k^*}$ and our algorithm terminates. We do not construct an explicit recursive partitioning algorithm that provably terminates with an identification of $\mathrm{Thin}(\Theta)$, however, we conjecture that any reasonable partitioning scheme will do so. Some schemes might be more efficient than others though.

We note here that the correctness of our dynamic programming is not compromised if the thinning operation identifies *any* subset of Θ such that Eqn. 15 holds, rather than the smallest such subset. Thus we envision variations on all of the thinning schemes already presented which might lead to less efficient representations of our potential functions, but involve less computational cost overall. For instance, one might consider a variation on the recursive partitioning scheme above in which, on each rectangle of the some *a priori* fixed partition, we remove all kernels dominated by other kernels. The union of all remaining associated parameters then serves as our proxy for $\mathrm{Thin}(\Theta)$.

It is possible that a less fastidious approach to thinning might also produce acceptable results. Many applications of dynamic programming employ a "beam search" strategy in which unpromising hypotheses are pruned at each stage of the computation, thus keeping the number of hypotheses manageably small. While this sacrifices the guaranteed identification of the globally optimal solution, such a technique often gives nearly optimal or good enough results. In this way, rather than performing the thinning operation as originally defined, we could simply take

$$\mathrm{Thin}'_M(\Theta) = \{\theta^1, \ldots, \theta^M\}$$

where $\Theta = \{\theta^1, \ldots, \theta^N\}$ is ordered so that $h(\theta^n) \geq h(\theta^{n+1})$ for $n = 1, \ldots, N-1$. The supposition here is that parameter values, θ, with relatively low values of $h(\theta)$ are unlikely to be part of the optimal parse and need not be considered further.

As a final comment, while thinning is employed in our definition of the multiplication operation in Section 2.3, it need not be performed there. The size of the representation of the H_C can be controlled as long as thinning is performed at regular intervals — say at each iteration of the algorithm. In some cases in might be easier to only perform thinning after "maxing out" since then we deal with lower dimensional potential functions.

4 An Example: Rhythmic Parsing

We now apply our methods to a problem encountered in music information retrieval: Rhythmic Parsing.

Rhythm is the aspect of music that deals with *when* events occur. Typically, rhythm in Western music is notated in a way that expresses the position of each note as a rational number, usually in terms of some relatively small common denominator. For instance, if we use the *measure* as our unit of notated position,

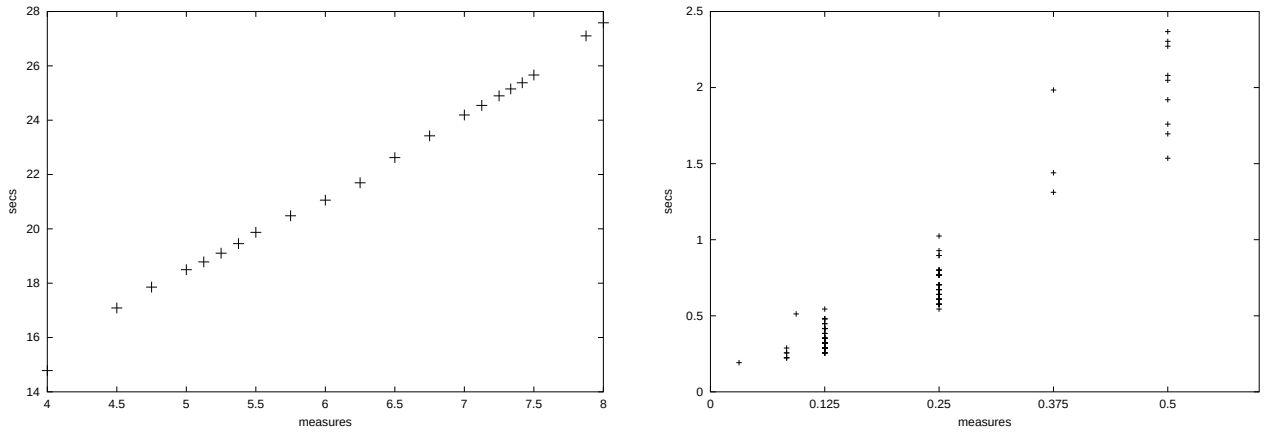

Figure 1: **Left:** Real time (seconds) vs. Musical time (measures) for a musical excerpt. **Right:** The actual durations (seconds) of notes grouped by the musical duration (measures).

then the sequence of measure positions $m_0 = 0, m_1 = 1/4, m_2 = 1, \ldots$ expresses the notion that the first note occurs at the beginning of the 1st measure, the second note occurs $1/4$ the way through the 1st measure, the third note occurs at the beginning of the 2nd measure, etc. How do these abstract positions relate to actual time? If the music is performed with mechanical precision then a *tempo* will map the measure positions to actual times. For instance, if the tempo is 3 seconds per measure, then the notes would occur at 0 secs, 3/4 secs, 3 secs, etc. However, such a performance would be nearly impossible for the human perform to create, and, moreover, would be undesirable. Much of the expressively quality of a musical performance comes from the way in which the actual note times deviate from what is prescribed by a literal interpretation of the printed music. In particular, there are two primary components to this *expressive timing* [8]. Firstly, the actual tempo is often not constant, but rather continually varied throughout the evolution of the performance. Secondly, there are more local (note by note) distortions which can be accidental, or can result from interpretive considerations.

Consider as an example the left panel of Figure 1 depicting a sequence of estimated note onset times $o_0, \ldots, o_N$, measured in seconds. These data were taken from a performance of an excerpt from Robert Schumann's *2nd Romance for Oboe and Piano*, estimated from the oboe part only. The audio file and the actual time estimates can be heard and examined at http://fafner.math.umass.edu/rhythmic_parsing. When these times (in seconds) are plotted against the known score positions (in measures), the points trace out a curve showing the player's rhythmic interpretation, as in the left panel of Figure 1. For instance, one can see the tempo variations as changes in the slope of the curve in this figure. Suppose, however, that the score positions are unknown, and we wish to estimate these positions from the observed times. This is the problem of *rhythmic parsing* which we consider here. Rhythmic parsing has applications in automatic score generation, in which a performance is converted to a musical score, music information retrieval, in which it is a basis for cataloging musical data for automatic search and analysis, and in musicology, in which it can be used to automatically separate notated rhythm and expressive timing. Variations of this problem have been addressed by [9], [10], [8], [11],[12].

Nearly every commercial music-writing program provides a mechanism to perform rhythmic parsing, most often by *quantizing* the observed note lengths, or more precisely inter-onset intervals (IOIs), to their closest note values (eighth note, quarter note, etc.), given a known tempo, or quantizing the observed note onset times to the closest points in a rigid grid [13]. While such quantization schemes can work reasonably well when the music is played with robotic precision (often a metronome is used), they perform poorly when faced with the more expressive and less accurate playing typically encountered. Consider the right panel of Figure 1 in which we have plotted the written note lengths in measures $(m_{n+1} - m_n)$ versus the actual note lengths (IOIs) in seconds $(o_{n+1} - o_n)$ from our musical excerpt. The large degree of overlap between the empirical distributions of each note length class demonstrates the futility of assigning note lengths through note-by-note quantization in this example. We propose, instead, a model based approach in which we define a probabilistic model (a CG distribution) on a rhythm process, a tempo process, and an observable process.

We then cast the rhythmic parsing problem as one of MAP estimation.

4.1 The Model

Suppose we have a set, $\mathcal{P}$, composed of the possible *measure positions* a note can occupy. In our example, the set was taken from the actual score to be

$$\mathcal{P} = \left\{ \frac{0}{1}, \frac{1}{8}, \frac{1}{4}, \frac{1}{3}, \frac{3}{8}, \frac{5}{12}, \frac{15}{32}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8} \right\}$$

however, a more generic $\mathcal{P}$ could be used when there is less available information about the data to be recognized. Let $P_0, P_1, \ldots, P_N$ be the discrete measure position process, $P_n \in \mathcal{P}, n = 0, \ldots, N = 128$. In interpreting these positions we assume that each consecutive pair of positions corresponds to a note length of at most one measure. For instance, with $\mathcal{P}$ given above, $P_n = 0/1, P_{n+1} = 1/4$ would mean the nth begins at the start of the measure and lasts for one quarter of a measure, while $P_n = 0/1, P_{n+1} = 0/1$ would mean the nth note begins at the start of the measure and lasts for one whole measure. We can then use

$$l(p_n, p_{n+1}) = \begin{cases} p_{n+1} - p_n & \text{if } p_{n+1} > p_n \\ 1 + p_{n+1} - p_n & \text{otherwise} \end{cases}$$

to unambiguously represent the length, in measures, of the transition from p_n to p_{n+1}. Thus, if $p_0, p_1, \ldots, p_N$ is known, we assign a score position, m_n, to every observation o_n by $m_n = p_0 + \sum_{\nu=1}^{n} l(p_\nu, p_{\nu-1})$. We model the P process as a time-homogeneous Markov chain with initial distribution

$$I(p_0) = \text{Prob}(P_0 = p_0)$$

and transition probability matrix

$$R(p_n, p_{n+1}) = \text{Prob}(P_{n+1} = p_{n+1} | P_n = p_n)$$

The time-varying tempo is the most important link between the printed note lengths, $l(P_n, P_{n+1})$, and the observed note lengths, $o_{n+1} - o_n$. Let $T_1, T_2, \ldots, T_N$ be the continuously-valued tempo process, measured in seconds per measure, which we model by

$$T_1 \sim N(\nu, \phi^2)$$

and

$$T_n = T_{n-1} + \delta_n$$

for $n = 2, 3, \ldots, N$ where $\delta_n \sim N(0, \tau^2 l(P_{n-1}, P_n))$. This model captures the property that the tempo tends to vary gradually and that longer notes allow for greater changes in local tempo.

Finally we assume that the observed note lengths $y_n = o_n - o_{n-1}$ for $n = 1, 2, \ldots, N$ are modeled by random variables

$$Y_n = l(P_{n-1}, P_n) T_n + \epsilon_n$$

where $\epsilon_n \sim N(0, \rho^2 l(P_{n-1}, P_n))$. Thus Y_n is a noisy observation of the length (seconds) predicted by the note length, $l(P_{n-1}, P_n)$ (measures), and the local tempo, T_n (seconds per measure). Note that our model assumes that longer notes lead to greater observation variance. These modeling assumptions lead to a graphical model whose directed acyclic graph is given in Figure 2.

The joint distribution of $P_0, \ldots, P_n$, $T_1, \ldots, T_N$, $Y_1, \ldots, Y_N$ given above is a CG distribution.

4.2 Identifying the Most Likely Rhythmic Parse

We begin by constructing the graph, $\bar{\mathcal{G}}$, for the conditional distribution of the unobserved variables, $P_0, \ldots, P_N$, $T_1, \ldots, T_N$ given $Y_1 = y_1, \ldots, Y_N = y_N$ according the procedure described in Section 2.1. To moralize the graph we need only add the edges (P_0, T_1), (P_1, T_1) and (P_{n+1}, T_n), $n = 1 \ldots, N-1$. We then restrict the graph by eliminating the nodes corresponding to the observations $\{Y_n\}$ and the edges adjacent to these nodes. The resulting graph is already triangulated without having to add additional edges and is shown

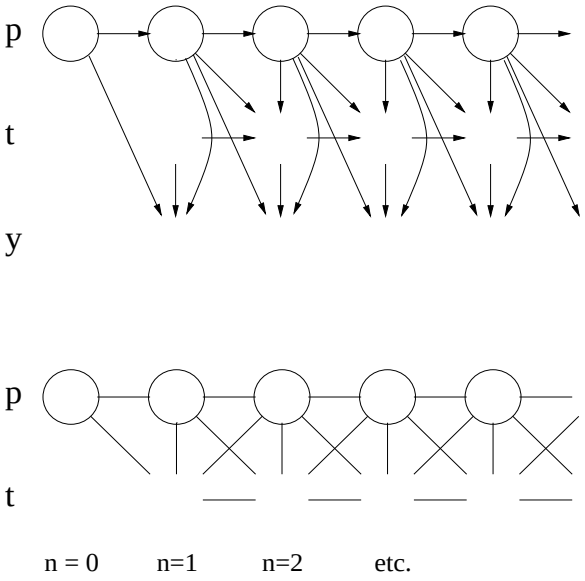

Figure 2: **Top:** The DAG, $\mathcal{G}$, describing the dependency structure of the variables in our rhythmic parsing example. Circles represent discrete variables while squares represent continuous variables. **Bottom:** The undirected graph, $\bar{\mathcal{G}}$, corresponding to the conditional distribution of position and tempo given the observations.

in the bottom panel of Figure 2. No further edges need be added to account for the asymmetric roles of discrete and continuous variables. There are N cliques of this graph, $C_1, \ldots, C_N$ with $C_1 = \{P_0, P_1, T_1\}$, and $C_n = \{P_{n-1}, P_n, T_{n-1}, T_n\}$, $n = 2, \ldots, N$. The junction tree on these cliques has linear connectivity structure with C_n and C_{n+1} adjacent in the junction tree, $n = 1, \ldots, N-1$ with separator $S_n = C_n \cap C_{n+1} = \{P_n, T_n\}$. Note that each clique and each separator has both discrete and continuous variables.

Having observed $Y_1 = y_1, \ldots, Y_N = y_N$, the posterior likelihood function on the unobserved variables, $\bar{f}(p_0, \ldots, p_N, t_1, \ldots, t_N)$, factors into a product of potentials

$$\bar{f} = \prod_{n=1}^{N} \phi_{C_n}$$

In particular

$$\begin{aligned} \phi_{C_1}(p_0, p_1, t_1) &= I(p_0) R(p_1, p_1) N(t_1; \nu, \phi^2) N(y_1; l(p_0, p_1) t_1, \rho^2 l(p_0, p_1)) \\ &= K(t_1; \theta_1(p_0, p_1)) \end{aligned}$$

where $N(\cdot\,; \mu, \sigma^2) = K(\cdot\,; (2\pi\sigma^2)^{-1/2}, \mu, 1/\sigma^2)$ is the univariate normal density function and $\theta_1(p_0, p_1)$ is computed by representing the normal density involving both t_1 and y_1 using Eqn. 24, eliminating y_1 from the same density using Eqn. 22, and multiplying with the other normal density using Eqn. 18. Similarly,

$$\begin{aligned} \phi_{C_n}(p_{n-1}, p_n, t_{n-1}, t_n) &= R(p_{n-1}, p_n) N(t_n; t_{n-1}, \tau^2 l(p_{n-1}, p_n)) N(y_n; l(p_{n-1}, p_n) t_n, \rho^2 l(p_{n-1}, p_n)) \\ &= K(t_{n-1}, t_n; \theta_n(p_{n-1}, p_n)) \end{aligned}$$

with $\theta_n(p_{n-1}, p_n))$ computed by representing the normal density involving t_n and t_{n-1} using Eqn. 24, eliminating y_n in the other normal density using Eqn. 22, extending this density to a potential involving both t_n and t_{n-1} using Eqn. 23 and multiplying the two together using Eqn. 18.

We arbitrarily choose C_N to be the root of the junction tree. Then $C_n \stackrel{S_{n-1}}{\rightarrow} C_{n-1}$ for $n = 2, \ldots, N$ and the dynamic programming recursion of Eqn. 7 becomes

$$H_{C_n}(x_{C_n}) = \phi_{C_n}(x_{C_n}) \max_{C_{n-1} \setminus S_{n-1}} H_{C_{n-1}}(x_{C_{n-1}})$$

$n = 2, \ldots, N$ or more explicitly

$$\begin{aligned} H_{C_2}(p_1, p_2, t_1, t_2) &= \phi_{C_2}(p_1, p_2, t_1, t_2) \max_{p_0} \phi_{C_1}(p_0, p_1, t_1) \\ H_{C_n}(p_{n-1}, p_n, t_{n-1}, t_n) &= \phi_{C_n}(p_{n-1}, p_n, t_{n-1}, t_n) \max_{p_{n-2}, t_{n-2}} H_{C_{n-1}}(p_{n-2}, p_{n-1}, t_{n-2}, t_{n-1}) \end{aligned}$$

$n = 3, \ldots, N$. Clearly these computations can be performed using the methodology we have presented.

4.3 Experiments

The parameters of our model are $I, R, \nu, \phi^2, \tau^2, \rho^2$, and of these parameters R is the most crucial. The two most extreme choices for R are the uniform transition probability matrix

$$R^{\text{unif}}(p_i, p_j) = 1/|\mathcal{P}|$$

and the matrix ideally suited to our particular recognition experiment

$$R^{\text{ideal}}(p_i, p_j) = \frac{|\{n : P_n = p_i, P_{n+1} = p_j\}|}{|\{n : P_n = p_n\}|}$$

R^{ideal} is unrealistically favorable to our experiments since this choice of R is optimal for recognition purposes and incorporates information normally unavailable; R^{unif} is unrealistically pessimistic in employing no prior information whatsoever. The actual transition probability matrices used in our experiments were convex combinations of these two extremes

$$R = \alpha R^{\text{ideal}} + (1 - \alpha) R^{\text{unif}}$$

for various constants $0 < \alpha < 1$. A more intuitive description of the effect of a particular α value is the *perplexity* of the matrix it produces: $\text{Perp}(R) = 2^{H(R)}$ where $H(R)$ is the $\log_2$ entropy of the corresponding Markov chain,

$$H(R) = -\sum_{p, p' \in \mathcal{P}} q_R(p) R(p, p') \log_2 R(p, p')$$

and q_R is the limiting distribution of the Markov chain. Roughly speaking, if a transition probability matrix has perplexity M, the corresponding Markov chain has the same amount of "indeterminacy" as one that chooses randomly from M equally likely possible successors for each state. The extreme transition probability matrices have

$$\begin{aligned} \text{Perp}(R^{\text{ideal}}) &= 1.92 \\ \text{Perp}(R^{\text{unif}}) &= 11 = |\mathcal{P}| \end{aligned}$$

In all experiments we chose our initial distribution, $I(p_0)$, to be uniform, thereby assuming that all starting measure positions are equally likely. The remaining constants, $\nu, \phi^2, \tau^2, \rho^2$ were chosen to be "reasonable" values.

The dynamic programming recursion involves the computation of

$$\begin{aligned} H_{S_1}(p_1, t_1) &\stackrel{\text{def}}{=} \max_{p_0} \phi_{C_1}(p_0, p_1, t_1) \\ H_{S_n}(p_n, t_n) &\stackrel{\text{def}}{=} \max_{p_{n-1}, t_{n-1}} H_{C_n}(p_{n-1}, t_{n-1}, p_n, t_n) \end{aligned}$$

for separators $\{S_n : n = 2 \ldots, N\}$. The computational feasibility of our approach relies on the representation of the $\{H_{S_n}\}$, and hence the $\{H_{C_n}\}$ staying manageably small. We use $\text{Size}(n)$ to denote the number of

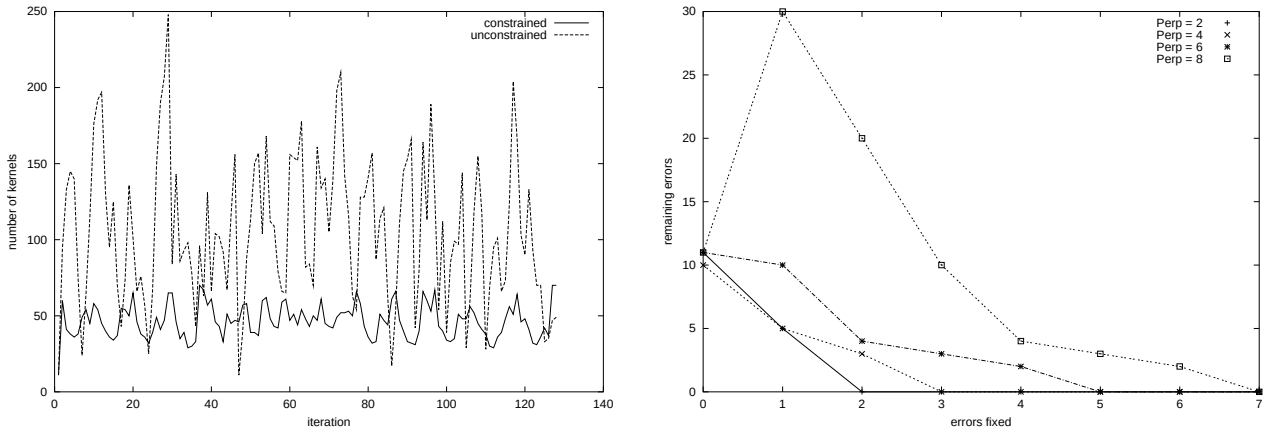

Figure 3: **Left:** The number of Gaussian kernels necessary to represent our potential functions, as a function of n. **Right:** The number of errors produced by our system at different perplexities and with different numbers of errors already corrected.

kernels used in the representation of H_{S_n}. The left panel of Figure 3 shows, with $\mathrm{Perp}(R) = 4$, the evolution of $\mathrm{Size}(n)$ under two different experimental conditions. The first uses the original one-dimensional thinning algorithm, $\mathrm{Thin}(\Theta)$, as explained in Section 3. The second uses the constrained version of the one-dimensional thinning algorithm, $\mathrm{Thin}_D(\Theta)$ in which the computation is confined to the interval $D = (1, 5)$ seconds per measure. This corresponds to a rather wide tempo range of 48 to 240 beats per minute (the composer's tempo marking is 104 beats per minute). In both cases it is evident that the complexity of the representation of the $\{H_{S_n}\}$ functions does not grow over the course of the calculations; we anticipate that this behavior is typical, and not particular to this example. In addition, it is evident that a significant computational savings resulted from the use of constrained thinning. The average number of kernels used in the representation of $H_{S_n}(\cdot, p_n)$, $\sum_{n=1}^{N} \mathrm{Size}(n)/(N \times |\mathcal{P}|)$, was 4.22 in the constrained case and 9.59 in the unconstrained case.

The rhythmic parsing problem is occasionally ambiguous even for musical experts, thus we expect that all parsing algorithms will make errors on some data sets. However, in many applications nearly perfect rhythmic parses are needed. Therefore, a usable algorithm should seek to label as much of the data correctly as possible, and be able to improve the labeling when mistakes are pointed out by a person. Our algorithm seems particularly well-suited to this kind of approach, since any post-process corrections can be treated as fixed variables in subsequent runs. The right panel of Figure 3 shows results of such a sequence of interactive parses. At four different perplexities, $\mathrm{Perp}(R) = 2, 4, 6, 8$, and with constrained thinning as above, we used our algorithm to identify the MAP configuration of the unobserved variables, $\{P_n, T_n\}$, and then iterated 7 times the procedure of identifying the first parse error and rerunning the algorithm using the new information. Thus the kth point in a trace of the plot shows the number of parse errors after a total of k corrections have been made. In all cases we see that the error is decreased to 0 after several corrections are made. This behavior results from the highly context-dependent nature of rhythmic parsing. Thus a single correction offers valuable information about neighboring note labels and can decrease the number of errors significantly.

5 Discussion

Many potential applications of graph-based probabilistic reasoning involve networks containing both discrete and continuous variables thereby underscoring the importance of computational machinery for performing inference in such mixed networks. Lauritzen's scheme handles the computation of local posterior marginals in the same domain of CG distributions we address here. One could, in principle, use either methodology for estimating unobserved variables, however, the two approaches are not equally applicable to all problems.

From a computational standpoint, the Achilles' heel of Lauritzen's scheme is its reliance on a junction tree with a *strong* root: If C_1 and C_2 are cliques in the junction tree with C_2 further from the root than C_1, then either $C_1 \cap C_2 \subseteq \Delta$ or $C_2 \backslash C_1 \subset \Gamma$ [3]. Junction trees with strong roots arise from *decomposable* marked graphs

[3]: These are triangulated graphs of discrete and continuous variables characterized as follows: If a path through the graph begins and ends with discrete nodes and otherwise is composed of continuous nodes, then the discrete nodes must be neighbors [14]. While one can always add edges to produce a decomposable marked graph, hence a junction tree with strong root, the result can easily become computationally intractable when the "interface" between discrete and continuous variables is large. For example, consider the graph of our experiments depicted in the bottom panel of Figure 2. The path $P_i, T_{i+1}, \ldots, T_j, P_j$ is contained in the graph $\bar{\mathcal{G}}$ for any $0 \leq i < j \leq N$. Hence, if a strong root is desired, each pair of discrete nodes must be connected and the entire collection of discrete nodes must be contained in a single clique. Clearly the ensuing computations are hopeless. In contrast, our scheme does not require a strong root and remains tractable in graphs having large interface between discrete and continuous variables, as illustrated by our example.

As discussed in Section 3, our scheme has a computational weakness too. To construct a MAP one must perform the thinning operation, since otherwise the representation of our H_C functions grows exponentially during the computation. Thinning is simple to perform when there is at most one continuous variable in the representation being thinned. We also sketched a thinning algorithm for arbitrarily dimensioned clique potentials, however the computational cost of this latter algorithm might be significant. Since the ultimate goal of thinning is computational feasibility of the algorithm, in some applications one might benefit by relaxing the goal of "minimal" thinning, and instead seeking to simply reduce the potential representation at each iteration, as discussed in Section 3.2. Also mentioned in this section was the possibilty of a computationally inexpensive approach to thinning (Thin$'$) that is no longer guaranteed to produce globally optimal estimates. These choices all have their strengths and weaknesses and we anticipate that the best solution to this computational issue will vary from application to application. While we expect that thinning will not represent a major stumbling block in the application of our methodology, it is remains a matter of some concern.

In reality, the two methods should really not be considered "competitors" since they address different problems. The posterior marginals computed by Lauritzen's scheme are useful when one cares about the mostly likely assignment to a *particular* variable: Is the *disease* present? Given an observation, what is the most likely *class*? Posterior marginals are not suitable for all inference problems, however. For instance, if in our rhythmic parsing example we computed the sequence of posterior modes,

$$p_n^* = \arg\max_{p_n \in \mathcal{P}} \text{Prob}(P_n = p_n | Y_1 = y_1, \ldots, Y_n = y_N)$$

$n = 0, \ldots, N$, then the estimated sequence $p_0^*, \ldots, p_N^*$ might well have

$$\text{Prob}(P_0 = p_0^*, \ldots, P_N = p_N^* | Y_1 = y_1, \ldots, Y_n = y_N) = 0$$

That is, while each p_n^* might be a reasonable estimate for P_n there is no guarantee that the sequence $p_0^*, \ldots, p_N^*$ is reasonable, or even makes sense, for $P_0, \ldots, P_N$. Partly for this reason we instead posed the inference problem as one of MAP estimation. Since a MAP is the most likely *configuration* of unobserved variables, MAP estimates are ideal when the interpretation one seeks is modeled as a such a configuration. Rhythmic parsing, speech recognition, image reconstruction, and soft decoding are examples of the many problems in which the desired interpretation corresponds to a configuration of variables.

It should be noted that the two approaches might best be served by collaboration rather than competition. In our example, the model parameters were set by hand, rather than estimated from training data since we had no methodology to perform training. In problems where Lauritzen's scheme is computationally tractable, the computation of posterior marginals could form the backbone of an EM-based method for estimating model parameters in CG distributions with incomplete observations, since EM depends on these marginals. Having improved a model during such a training phase, one could then estimate a configuration of unobserved variables via the MAP estimation methodology we have presented.

6 Gaussian Kernels

A Gaussian kernel is a multivariate function of the form of Eqn. 10. The following identities hold for such functions. The derivations of these results are quite straightforward and are not included here.

Multiplication:

$$K(x; h_1, m_1, Q_1) K(x; h_2, m_2, Q_2) = K(x; h, m, Q) \tag{18}$$

where

$$\begin{aligned} h &= h_1 h_2 e^{-\frac{1}{2}(m_1^t Q_1 m_1 + m_2^t Q_2 m_2 - m^t Q m)} \\ m &= Q^-(Q_1 m_1 + Q_2 m_2) \\ Q &= Q_1 + Q_2 \end{aligned}$$

where Q^- is the *generalized inverse* of Q [16], [17]. We deal here only with nonnegative definite symmetric matrices; in this case Q^- can be expressed as

$$Q^- = UD^-U^t$$

where $Q = UDU^t$ with U unitary and D diagonal, and D^- is diagonal with

$$D_{ii}^- = \begin{cases} 1/D_{ii} & D_{ii} > 0 \\ 0 & D_{ii} = 0 \end{cases}$$

Maxing Out: Let m and Q be partitioned as

$$m = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \tag{19}$$

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \tag{20}$$

Then

$$\max_{x_2} K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) = K(x_1; h, m_1, \tilde{Q}) \tag{21}$$

where

$$\tilde{Q} = Q_{11} - Q_{12}Q_{22}^-Q_{21}$$

In the event that x_1 has *no* components, we have maximized over all variables of the kernel and interpret $K(x_1; h, m_1, \tilde{Q})$ as the constant h.

Fixing Variables: Let m and Q be partitioned as in Eqns. 19,20. Regarding x_2 as fixed

$$K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) = K(x_1; \tilde{h}, \tilde{m}, \tilde{Q}) \tag{22}$$

where

$$\begin{aligned} \tilde{h} &= h e^{-\frac{1}{2}(x_2 - m_2)^t (Q_{22} - Q_{21}Q_{11}^- Q_{12})(x_2 - m_2)} \\ \tilde{m} &= m_1 - Q_{11}^- Q_{12}(x_2 - m_2) \\ \tilde{Q} &= Q_{11} \end{aligned}$$

Extension: The kernel $K(x_1; h, m, Q)$ can be viewed as a function of x_1 and x_2 by

$$K(x_1; h, m, Q) = K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, \begin{pmatrix} m \\ 0 \end{pmatrix}, \begin{pmatrix} Q & 0 \\ 0 & 0 \end{pmatrix}) \tag{23}$$

Conditional Gaussian Densities: If $x_2 = \alpha^t x_1 + \beta + \xi$ where x_2 is univariate and $\xi \sim N(\mu, \sigma^2)$, the conditional density of x_2 given x_1 is

$$f_v(x_2|x_1) = K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) \tag{24}$$

where

$$\begin{aligned} h &= (2\pi\sigma^2)^{-1/2} \\ m &= \begin{pmatrix} 0 \\ \beta + \mu \end{pmatrix} \\ Q &= \frac{1}{\sigma^2} \begin{pmatrix} \alpha\alpha^t & -\alpha \\ -\alpha^t & 1 \end{pmatrix} \end{aligned}$$

References

[1] Rabiner L. (1989), "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* 77, 257–286, 1989.

[2] Dawid A. P. (1992) "Applications of a General Propagation Algorithm for Probabilistic Expert Systems," *Statistics and Computing*, vol. 2, 25-36.

[3] Cowell R. G., Dawid A. P., Lauritzen S. L., Spiegelhalter D. J. (1999), "Probabilistic Networks and Expert Systems," Springer, New York, NY.

[4] Lauritzen S. L. and Wermuth N (1984), "Mixed Interaction Models," *Technical Report R-84-8*, Institute for Electronic Systems, Aalborg University.

[5] Lauritzen S. L. and Wermuth N (1989), "Graphical Models for Associations Between Variables, some of which are Qualitative and some Quantitative," *Annals of Statistics*, 17, 31-57.

[6] Lauritzen S. L. (1992), "Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models," *Journal of the American Statistical Association*, Vol. 87, No. 420, (Theory and Methods), pp. 1098–1108.

[7] Lauritzen S. L. and F. Jensen (1999), "Stable Local Computation with Conditional Gaussian Distributions," *Technical Report R-99-2014*, Department of Mathematic Sciences, Aalborg University.

[8] Desain P., Honing H. (1989), "The Quantization of Musical Time: A Connectionist Approach," *Computer Music Journal*, Vol 13, no. 3.

[9] Cemgil A. T., Kappen B., Desain P., Honing, H. (2000), "On Tempo Tracking: Tempogram Representation and Kalman Filtering" *Proceedings of the International Computer Music Conference*, Berlin, 2000.

[10] Desain P., Honing H. (1994), "A Brief Introduction to Beat Induction," *Proceedings of the International Computer Music Conference*, San Francisco, 1994.

[11] Desain P., Aarts R., Cemgil A. T., Kappen B., van Thienen H, Trilsbeek P. (1999), "Robust Time-Quantization for Music from Performance to Score," *Proceedings of 106th Audio Engineering Society conference*, May 1999, Munich.

[12] Cemgil A. T., Desain P., Kappen B. (2000), "Rhythm Quantization for Transcription," *Computer Music Journal*, Vol. 24:2,60-76.

[13] Trilsbeek P., van Thienen H., (1999), "Quantization for Notation: Methods used in Commercial Music Software," handout at *106th Audio Engineering Society conference*, May 1999, Munich.

[14] Leimer H.-G. (1989), "Triangluated Graphs with Marked Vertices," *Annals of Discrete Mathematics*, 41, 311-24.

[15] Lauritzen S. L. (1996), "Graphical Models," Clarendon Press, Oxford.

[16] Rao C. R., Mitra S. K., (1971), "Generalized Inverse of Matrices and its Applications," John Wiley and Sons, New York.

[17] Graybill, F., (1969), "Matrices with Applications in Statistics," Wadsworth International Group, Belmont, CA.