MARKOV RANDOM FIELDS AND MAXIMUM ENTROPY MODELING FOR MUSIC INFORMATION RETRIEVAL

Jeremy Pickens and Costas Iliopoulos
Department of Computer Science
King's College London*
London WC2R 2LS, England
jeremy,csi@dcs.kcl.ac.uk

ABSTRACT

Music information retrieval is characterized by a number of various user information needs. Systems are being developed that allow searchers to find melodies, rhythms, genres, and singers or artists, to name but a few. At the heart of all these systems is the need to find models or measures that answer the question "how similar are two given pieces of music". However, *similarity* has a variety of meanings depending on the nature of the system being developed. More importantly, the features extracted from a music source are often either single-dimensional (i.e.: only pitch, or only rhythm, or only timbre) or else assumed to be orthogonal. In this paper we present a framework for developing systems which combine a wide variety of non-independent features without having to make the independence assumption. As evidence of effectiveness, we evaluate the system on the polyphonic theme similarity task over symbolic data. Nevertheless, we emphasize that the framework is general, and can handle a range of music information retrieval tasks.

Keywords: Random Fields, Music Modeling

1 INTRODUCTION

Recent interest in the area of music information retrieval and related technologies is exploding. Digital music collections are becoming available locally (computer hard disks, mp3 players) and remotely (online music stores, digitial libraries). However, few of the existing techniques take advantage of recent developments in statistical modeling. In this paper we discuss an application of Markov Random Fields to the problem of creating accurate yet flexible statistical models of music. With such models in hand the challenges of developing effective searching, browsing and organization techniques for the growing bodies of music collections may be successfully met.

Random Fields are a generalization of Markov chains to multi-dimensional spatial processes. They are incredibly flexible, allowing us to model arbitrary interactions between variables. They have been very popular in modeling of physical systems, and recently have demonstrated superior performance in a number of language-related applications such as Berger et al. (1996), Della Pietra et al. (1997) and Lafferty et al. (2001). Finally, and perhaps most importantly, random fields are extremely attractive from a theoretical standpoint. Probability distributions over the fields have exponential form, consistent with the maximum-entropy framework for inference. The objective function is globally $\cap$-convex with respect to parameters of the model, and so parameters can be learned effectively through iterative methods. Furthermore, there exists a principled way of inducing the structure of the field that guarantees improvement in the objective function, and in some cases allows closed-form solutions.

2 MUSIC REPRESENTATION

Music has several possible representations. In its most unstructured form, music can be represented as a sequence of audio signal samples, as for example in a wav or mp3 file. On the other end of the spectrum, music may be represented as instructions to a performer, as in sheet music. In the middle are representations such as MIDI, in which the onset, duration, and pitch of every note in a piece of music is known but no other information is necessarily available. Not to be overlooked, music may also be represented using metadata, such as song title, artist, genre, year, instrumentation and so on. For the evaluation of our current model, we will focus on symbolic pitch information. But we emphasize that the model in general can accomodate features from any representation, including both content- and metadata-related features, simultaneously, as long as the feature is the answer to a binary question. This will be covered in more detail at the end of the paper.

In polyphonic music, we may think of the notes as a two-dimensional graph, with ticks (time) along the x-axis, and MIDI note number (1 to 128) along the y-axis. At any point along the y-axis, notes turn "on", remain on for a particular duration, and then turn back "off" again. As a quick example, see the figures below. Black circles

*This work is supported by EPSRC grant GR/N63123/01

represent notes being on. White circles represent notes being off.

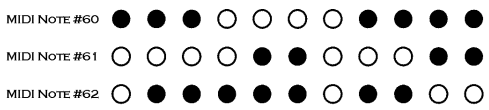


Figure 1: Example music sequence, preprocessed

For our current purposes (though not necesssary in general) we make the following simplifications. We begin by selecting only the onset times of each new pitch in the sequence, ignoring the duration of the note. Next, we remove all onset times which contain no pitches. (Thus, we are throwing away not only the duration of the notes themselves, but the duration between notes, including rests) Finally, we reduce the 128-note y-axis to a 12-note octave-invariant pitch set by taking the mod-12 value of every symbolic pitch number. Our example above becomes:

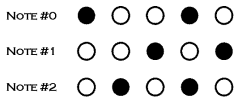


Figure 2: Example music sequence, postprocessed

3 MARKOV CHAIN RETRIEVAL MODEL

In the previous section we showed how polyphonic music can be represented as a temporal progression of 12-dimensional binary vectors. In this section we introduce as a baseline approach a simple modeling and retrieval system based on extracting multiple, sometimes overlapping Markov chains from this sequence.

A Markov chain model proceeds in three stages. In the first stage, we extract the chains (or *features*). In the second stage, we use the relative frequency counts of these extracted chains to estimate a state transition distribution. In the final stage, we compare models estimated from both a music query and a collection document and rank collection documents by this similarity score.

For feature extraction, we simply count the number of length m sequences through a piece of music. This approach is broadly similar to that of Doraisamy and Rüger (2001). For example, if we extract chains of length m=2 from the music in Figure 2, we end up with the following set of sequences: {0⇝2, 2⇝1, 1⇝0, 1⇝2, 0⇝1, 2⇝1}.

In the second stage, we divide these chains into two parts, the "previous state", or history, and the "current state". We define the history H as the first m-1 notes in the sequence, and the current state n as the last note in the sequence. For example, with an m=2 chain "1⇝2", the history is the state "1" and the current state is "2". With an m=3 chain "1⇝2⇝1", the history is the state "1⇝2" and the current state is "1"

We next obtain parameters for the conditional probability distribution $\hat{P}(n|H)$ by doing maximum likelihood estimation on the complete set of extracted chains, where $|H, n|$ is the number of times the chain with history H followed by note n is observed.

$$\hat{P}(n|H) = \frac{|H, n|}{\sum_{H_i} |H_i, n|} \quad (1)$$

Basic Dirichlet smoothing is used to overcome the zero frequency problem (Zhai and Lafferty). Figure 3 shows an (unsmoothed) example for a sample 3-state model trained with the data from Figure 2.

Counts			
	0	1	2
0	0	1	1
1	1	0	1
2	0	2	0

$\hat{P}(n\|H)$			
	0	1	2
0	0	0.5	0.5
1	0.5	0	0.5
2	0	1.0	0

Figure 3: Example 1^{st}-order ($m = 2$) state transition counts (*left*) and state transition distribution (*right*)

Prior to retrieval, at indexing time, we estimate $\hat{P}(n|H)$ for every piece of music in the collection. At retrieval time, when presented with a query, we estimate a model for the query in the exact same manner. Similarity is calculated between the query and every document in the collection using the Kullback-Leibler (KL) divergence measure. Documents are ranked by increasing dissimilarity to the query.

4 MARKOV RANDOM FIELD RETRIEVAL MODEL

In the previous section, we modeled polyphonic music as a sequence of Markov chains across neighboring 12-bit binary note vectors. However, there is one main problem with this method: Since the music is polyphonic, the extracted Markov chain features overlap. This overlap causes the model to count certain notes more than once, which has the effect of overestimating certain transitions. Using Markov chains makes the assumption that notes occuring at the same point in time are independent of each other; this is clearly not true. Therefore, we turn to a framework that allows us to selectively model the interactions (dependencies) between the individual notes.

4.1 Nature of the Field

Suppose we are given a musical piece represented by T simultaneities (binary vectors of length 12). With this piece we will associate a lattice of binary variables $\{n_{i,t}\}$, indexed by the time $t = 1 \ldots T$, and by note index $i = 0 \ldots 11$. Each variable $\{n_{i,t}\}$ can be either 0 or 1, indicating whether a note i is on or off at time t. Figure 4 illustrates the lattice. Our goal is to develop a model that will allow us to predict the value $n_{i,t}$ from the values of the surrounding variables. In other words, we would like to develop an estimate for the probability distribution $P(n_{i,t}|\{n_{j,s} : j \neq i \text{ or } s \neq t\})$. It is important to stress that we do not want to assume independence among the conditioning variables, or restrict the conditioning to the immediate neighbors of $n_{i,t}$. On the contrary, we believe that

$$
\begin{array}{ccccccc}
n_{0,1} & n_{0,2} & n_{0,3} & n_{0,4} & \dots & n_{0,t} & \dots \\
n_{1,1} & n_{1,2} & n_{1,3} & n_{1,4} & \dots & n_{1,t} & \dots \\
n_{2,1} & n_{2,2} & n_{2,3} & n_{2,4} & \dots & n_{2,t} & \dots \\
 & & & \dots & & & \\
n_{11,1} & n_{11,2} & n_{11,3} & n_{11,4} & \dots & n_{11,t} & \dots
\end{array}
$$

Figure 4: Variables of a "Musical" Markov Random Field

the value of $n_{i,t}$ is strongly influenced by both its short-range and long-range neighbors in the lattice. However, for the scope of this paper we will impose several limitations on what kind of dependencies may exist in our field.

The first limitation we impose concerns the temporal nature of music. In the most general formulation of a random field, the value of the note $n_{i,t}$ may be influenced by both the notes that precede it $\{n_{j,s \leq t}\}$, and notes that follow it $\{n_{j,s>t}\}$. For comparison with the chain model, in our initial random field model we will restrict the dependencies to only those notes that precede the target note in the sequence. For every note $n_{i,t}$ we define the concept of *history* or *neighborhood* $H_{i,t}$ to include the notes that either occur before time t, or notes that occur at time t, but have an index lower than i:

$$H_{i,t} = \{n_{j,s} : s < t\} \cup \{n_{j,s} : s = t, j < i\} \quad (2)$$

Notes in $H_{i,t}$ are the ones that can be examined when we are making the prediction regarding $n_{i,t}$. In other words, we assume that the probability of note i playing at time t is completely determined by $H_{i,t}$ in our model. We reiterate that we still allow arbitrary dependencies within the subset defined by the neighborhood $H_{i,t}$.

In general, we need not constrain our neighborhood in this manner. We believe that better models could be created by considering notes that will be played in the "future", not just in the past. However, we wish to do two things with this paper: (1) introduce random fields as a modeling framework suitable for music information retrieval, and (2) evaluate this framework by comparing it with a similar model (Markov chains). Markov chains only consider the past. If we were to introduce a version of the random field that considered the future as well as the past, then it would be unclear as to whether performance gains of fields over chains were due to inherent modeling superiority, or to the fact that the fields had use of more data. Thus, we constrain the field neighborhood in such a manner as to make it fair and analogous as possible to the chain models. Future work (no pun intended) need not adhere to this constraint.

4.2 Conjunctive Features

The second limitation we impose on the conditional probability $P(n_{i,t}|H_{i,t})$ concerns the nature of dependencies that will be modeled by a field. In general, a random field framework allows arbitrary dependencies (or *features*) between the target $n_{i,t}$ and its neighborhood $H_{i,t}$. For example, $n_{i,t}$ may depend on the answer to the following question: "*what is the total number of times note i was played in the history $H_{i,t}$?*". For the sake of simplicity and elegance we will deliberately restrict allowed

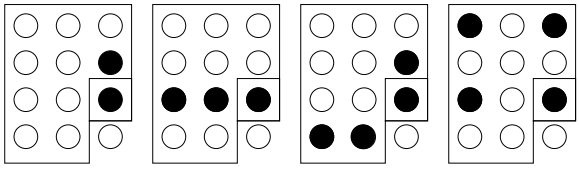


Figure 5: Examples of musical features that may be induced to predict the probability of note 2 being played at time t. Black circles represent notes that are part of the feature function. Boxed black circle denotes the note $n_{2,t}$. Boxed area represents the history $H_{2,t}$. From left to right, the features are: $\{n_{2,t}\ n_{1,t}\}$, $\{n_{2,t}\ n_{2,t-1}\ n_{2,t-2}\}$, $\{n_{2,t}\ n_{1,t}\ n_{3,t-1}\ n_{3,t-2}\}$, $\{n_{2,t}\ n_{0,t}\ n_{2,t-2}\ n_{0,t-2}\}$

dependencies to binary questions of the form: "*was note j played at time s before t?*". We will also allow generalizations where a question is asked about some subset S of the notes in the allowed history $H_{i,t}$. The answer to a question of this form will be called the feature function f_S, and S will be referred to as the *support* of f. For a given support $S \in H_{i,t}$, the feature function f_S is defined as the conjunction of answers about the individual notes in $n_{j,s} \in S$:

$$f_S(n_{i,t}, H_{i,t}) = n_{i,t} \prod_{n_{j,s} \in S} n_{j,s} \quad (3)$$

Defined in this manner, our feature functions are always boolean, and equal to 1 if all the notes defined by S were played before the target note $n_{i,t}$. A feature function always includes the target note $n_{i,t}$. This is not a fallacy in the model, since $n_{i,t}$ will never actually be considered a part of its own history. Presence of $n_{i,t}$ in the feature serves only to tie the occurrences of notes in S to the occurrence of $n_{i,t}$. If the feature is considered likely, that is evidence in favor of predicting $n_{i,t} = 1$. If the feature does not occur, it suggests that $n_{i,t}$ is likely to be zero.

One final comment: we choose to make features time-invariant, but not index invariant. This means that a feature is expected to characterize the same kind of dependency, regardless of the time index t of the target $n_{i,t}$, but that the feature is not (pitch) transposition invariant. Consequently, we will index the time component of the notes in S not in absolute values but relative to the time t. We do not do the same for the note index i, so these indices will remain absolute. As an illustration, Figure 5 contains some examples of features that could have an impact on note "2" at time t.

4.3 Exponential Form

At this point we are ready to select the parametric form that we will be using for computing the probabilities $P(n_{i,t}|H_{i,t})$. There are a number of different forms we could choose, but it turns out that for random fields there is a natural formulation of the distribution that is given by the maximum-entropy framework. Suppose we are given a set $\mathcal{F}$ of feature functions that define the structure of the field. The maximum-entropy principle states that we should select the parametric form that is: **(i)** consistent with the structure imposed by $\mathcal{F}$ and **(ii)** makes the least

amount of unwarranted assumptions — that is the most uniform of all distributions consistent with $\mathcal{F}$. The family of functions that satisfies these two criteria is the exponential (or log-linear) family, expressed as:

$$\hat{P}(n_{i,t}|H_{i,t}) = \frac{1}{Z_{i,t}} \exp\left\{\sum_{f\in\mathcal{F}} \lambda_f f(n_{i,t}, H_{i,t})\right\} \quad (4)$$

In equation (4), the set of scalars $\Lambda = \{\lambda_f : f \in \mathcal{F}\}$ is the set of Lagrange multipliers for the set of structural constraints $\mathcal{F}$. Intuitively, the parameter λ_f ensures that our model predicts feature f as often as it should occur in reality. $Z_{i,t}$ is the normalization constant that ensures that our distribution sums to unity over all possible values of $n_{i,t}$. In statistical physics, it is known as a *partition function* and is defined as follows:

$$Z_{i,t} = \sum_n \exp\left\{\sum_{f\in\mathcal{F}} \lambda_f f(n, H_{i,t})\right\} \quad (5)$$

For a general random field, the partition function $Z_{i,t}$ is exceptionally hard to compute, since it involves summation over all possible states of the system. In a typical system the number of states is exponential in the number of field variables, and direct computation of equation (5) is not feasible.

In our case, our assumption of no dependencies between the current state notes $n_{i,t}$ makes computation of the partition function extremely simple. Recall that all underlying field variables are binary, so equation (5) only needs to be computed for two cases: $n_{i,t} = 0$ and $n_{i,t} = 1$. We can further simplify the problem if we recall that every feature function f is a binary conjunction, and every f includes $n_{i,t}$ in its support. As a direct consequence, $f(n_{i,t}, H_{i,t})$ is non-zero only if $n_{i,t} = 1$. The assertion holds for all feature functions $f\in\mathcal{F}$, which implies that the summation inside the exponent in equations (4) and (5) is zero whenever $n_{i,t} = 0$. These observations allow us to re-write equation (4) in a form that allows very rapid calculations:

$$\begin{aligned} \hat{P}(n_{i,t} = 1|H_{i,t}) &= \sigma\left\{\sum_{f\in\mathcal{F}} \lambda_f f(1, H_{i,t})\right\} \\ \hat{P}(n_{i,t} = 0|H_{i,t}) &= 1 - P(n_{i,t} = 1|H_{i,t}) \end{aligned} \quad (6)$$

Here σ is the sigmoid function, defined as: $\sigma(x) = \frac{e^x}{1+e^x}$. We have stated equation (6) as a special case applicable to our particular setting. In the remaining arguments we will use the form given by equations (4) and (5) to ensure generality.

4.4 Objective Function

The ultimate goal of this project is to develop a probability distribution $\hat{P}(n_{i,t}|H_{i,t})$ that will accurately predict the notes $n_{i,t}$ in polyphonic music. There exist a number of different measures that could indicate the quality of prediction. We choose one of the simplest — log-likelihood of the training data. Given a training set $\mathcal{T}$ consisting of T simultaneities with 12 notes each, the log-likelihood is simply the average logarithm of the probability of producing every note in $\mathcal{T}$:

$$\begin{aligned} \mathcal{L}_{\hat{P}} &= \frac{1}{12T} \log \prod_{t=1}^{T} \prod_{i=0}^{11} \hat{P}(n_{i,t}|H_{i,t}) \\ &= \sum_H \tilde{P}(H) \sum_n \tilde{P}(n|H) \log \hat{P}(n|H) \end{aligned} \quad (7)$$

In the second step in equation (7) we re-expressed the log-likelihood in terms of the expected *cross-entropy* between the target distribution $\tilde{P}(n|H)$ and the estimate $\hat{P}(n|H)$ produced by our field. The target empirical distribution $\tilde{P}(n|H)$ can be computed directly from the training set $\mathcal{T}$, it is just the relative frequency of observing a note n together with the history H across all the positions (i, t) in the field:

$$\tilde{P}(n|H) = \frac{1}{12T} \sum_{t=1}^{T} \sum_{i=0}^{11} \delta(n, n_{i,t}) \delta(H, H_{i,t}) \quad (8)$$

Here δ refers to the Kronecker delta function, defined as:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Returning our attention to equation (7), we stress that the expectation $\sum_H[\ldots]$ is performed over all possible values that a history H of a note might take. This set is exponentially large, and a direct computation would be infeasible. However, for computation we always use the first part (top) of equation (7), whereas the second part (bottom) comes very handy in the algebraic derivations of the field induction algorithm.

4.5 Maximum Entropy

To summarize, in the previous two subsections we restricted ourselves to the exponential (Gibbs) form of the probability distribution $\hat{P}(n|H)$, and declared that our objective is to maximize the likelihood of the training data within that parametric form. It is important to note that there is a different statement of objectives that provably leads to exactly the same exponential solution $\hat{P}(n|H)$. Rather than focus on maximum-likelihood, we could search for the most uniform distribution $\hat{P}(n|H)$ that is consistent with the structure imposed by $\mathcal{F}$. To clarify what we mean by the structure consistency, suppose $f \in \mathcal{F}$ is a feature of the field. Let $\tilde{E}[f]$ denote the *empirical* or *target* expected value of f, which is simply how often the feature actually occurs in the training data $\mathcal{T}$:

$$\begin{aligned} \tilde{E}[f] &= \sum_H \tilde{P}(H) \sum_n \tilde{P}(n|H) f(n, H) \\ &= \frac{1}{12T} \sum_{t=1}^{T} \sum_{i=0}^{11} f(n_{i,t}, H_{i,t}) \end{aligned} \quad (10)$$

Similarly, our estimate $\hat{P}(n|H)$ gives rise to the *predicted* expectation $\hat{E}[f]$ for the function f. Predicted expected value is simply how often our model "thinks" that f should occur in the training set:

$$\hat{E}[f] = \sum_H \tilde{P}(H) \sum_n \hat{P}(n|H) f(n,H) \quad (11)$$
$$= \frac{1}{12T} \sum_{t=1}^{T} \sum_{i=0}^{11} \sum_n \hat{P}(n|H_{i,t}) f(n,H_{i,t})$$

The key difference between $\hat{E}[f]$ and $\tilde{E}[f]$ is that we do not look at the actual value $n_{i,t}$ when we compute $\hat{E}[f]$, instead we "predict" it from our model $\hat{P}(n|H)$. Given the two expectations in equations (10) and (11) it is natural to strive that they be equal, that is we'd like to arrange our model in such a way that predicted frequency $\hat{E}[f]$ of any feature f matches its actual frequency of occurrence $\tilde{E}[f]$. Furthermore, if there are multiple distributions $\hat{P}(n|H)$ that honor the constraint that $\hat{E}[f] = \tilde{E}[f]$, the maximum-entropy principle would guide us to pick the distribution that makes the least amount of assumptions about the data, or equivalently, maximizes its own expected entropy:

$$Ent_{\hat{P}} = \sum_H \tilde{P}(H) \sum_n \hat{P}(n|H) \log \hat{P}(n|H) \quad (12)$$

Curiously, maximizing the entropy subject to the constraint that $\tilde{E}[f] = \hat{E}[f]$ for every feature f turns out to be equivalent to assuming an exponential form for our probability distribution $\hat{P}(n|H)$ and maximizing the likelihood given by equation (7).

4.6 Feature Induction

In the previous sections we outlined the general structure of a random field over polyphonic music and stated our objective: to learn the probability distribution $\hat{P}(n|H)$ that maximizes the likelihood of the training data (equation (7)). Recall that we selected the exponential form for $\hat{P}(n|H)$. If we examine equation (4) we note that there are two things the model depends on. The first and the most important in our opinion is the structure of the field $\mathcal{F}$, represented as a set of constraints or feature functions $f \in \mathcal{F}$. These constraints represent most significant dependencies between the variables of the field. The second thing we learn is the set of weights $\Lambda = \{\lambda_f\}$, one for each feature $f \in \mathcal{F}$. We know that Λ and $\mathcal{F}$ are intimately intertwined and we need to learn them simultaneously, but for the sake of clarity we split the discussion in two sections. This section will describe how we can incrementally induce the structure $\mathcal{F}$ of the field, starting with a very flat, meaningless structure (primitive, atomic features) generalizing to more interesting complex relationships.

The field induction procedure closely follows the algorithm described in Della Pietra et al. (1997), the primary difference being that we are dealing with a conditional field, whereas Della Pietra uses a joint model. We start with a field that contains only individual notes, without any dependencies: $\mathcal{F}^0 = \{n_{i,t} : i = 0 \ldots 11\}$. We will incrementally update the structure $\mathcal{F}$ by adding the features g that result in the greatest improvement in the objective function. Suppose $\mathcal{F}^k = \{f_S\}$ is the current field structure. Also assume that the corresponding weights Λ^k are optimized with respect to $\mathcal{F}^k$. We would like to add to $\mathcal{F}^k$ a new feature g that will allow us to further increase the likelihood of the training data. In order to do that we first need to form a set of candidate features $\mathcal{G}$ that could be added. We define $\mathcal{G}$ to be the set of all one-note extensions of the current structure $\mathcal{F}$:

$$\mathcal{G} = \left\{ f_S \cdot n_{j,s} \quad : \quad \begin{array}{l} f_S \in \mathcal{F} \text{ and there exists} \\ n_{j',s'} \in S \text{ such that } |s - s'| \leq 2 \end{array} \right\} \quad (13)$$

In other words, we form new candidate features g taking an existing feature f and attaching a single note $n_{j,s}$ that is not too far from f in time (in our case, not more than by two simultaneities). Naturally, we do not include as candidates any features that are already members of $\mathcal{F}$. Now, following the reasoning of Della Pietra, we would like to pick a candidate $g \in \mathcal{G}$ that will result in the maximum improvement in the objective function. Suppose that previous log-likelihood based only on $\mathcal{F}^k$ was $\mathcal{L}_{\hat{P}}$. Now, if we add a feature g weighted by the multiplier α, the new likelihood of the training data would be:

$$\mathcal{L}_{\hat{P}+\{\alpha g\}} = \mathcal{L}_{\hat{P}} + \alpha \tilde{E}[g] - \log \hat{E}[e^{\alpha g}] \quad (14)$$

When we add a new feature g to the field, we would like to add it with a reasonable weight α, preferably the weight that maximizes the contribution of α. We can achieve that by differentiating the new log-likelihood $\mathcal{L}_{\hat{P}+\{\alpha g\}}$ with respect to α and finding the root of the derivative:

$$0 = \frac{\partial \mathcal{L}_{\hat{P}+\{\alpha g\}}}{\partial \alpha} \iff \alpha = \log \left[\frac{\tilde{E}[g](1 - \hat{E}[g])}{\hat{E}[g](1 - \tilde{E}[g])} \right] \quad (15)$$

An important observation to make is that we arrived at a closed-form solution for the optimal weight α to be assigned to the new feature g. The closed-form solution is a special property of binary feature functions, and greatly simplifies the process of inducing field structure. Knowing the optimal value of α in closed form allows us to compute the resulting improvement, or *gain*, in log-likelihood, also in closed form:

$$Gain = \tilde{E}[g] \log \frac{\tilde{E}[g]}{\hat{E}[g]} + (1 - \tilde{E}[g]) \log \frac{1 - \tilde{E}[g]}{1 - \hat{E}[g]} \quad (16)$$

The final form is particularly interesting, since it represents the Kullback-Leibler divergence between two Bernoulli distributions with expected values $\tilde{E}[g]$ and $\hat{E}[g]$ respectively.

4.7 Parameter Estimation

In the previous section we described how we can automatically induce the structure of a random field by incrementally adding the most promising candidate feature $g \in \mathcal{G}$. We also presented the closed form equations that allow us to determine the improvement in log-likelihood that would result from adding g to the field, and the optimal weight α that would lead to that improvement. What we did not discuss is the effect of adding g on the

weights of other features already in the field. Since the features $f \in \mathcal{F}$ are not independent of each other, adding a new feature will affect the balance of existing features. From equation (16) we know that the new log-likelihood $\mathcal{L}_{\hat{P}+\{\alpha g\}}$ is always going to be better than the old one $\mathcal{L}_{\hat{P}}$ (unless the field is saturated and cannot be improved anymore). However, this does not guarantee that the current set of weights Λ is optimal for the new structure. We may be able to further improve the objective by re-optimizing the weights for all functions that are now in the field.

Assume now that the structure $\mathcal{F}$ contains all the desired features. We would like to adjust the set of weights Λ, so that the objective function $\mathcal{L}_{\hat{P}}$ is maximized. This is accomplished by computing the partial derivatives of $\mathcal{L}_{\hat{P}}$ with respect to each weight $\lambda_{f'}$, with the intention of driving these derivatives to zero:

$$\frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_{f'}} = \tilde{E}[f'] - \hat{E}[f'] \tag{17}$$

Unfortunately, there is no closed-form solution that would allow us to set the weights to their optimal values. Instead, we utilize an iterative procedure that will drive the weights towards the optimum. There are a number of algorithms for adjusting the weights in an exponential model, the most widely known being the Generalized Iterative Scaling (GIS) algorithm proposed by Darroch and Ratcliff (1972). However, iterative scaling is extremely slow; much faster convergence can be achieved by using variations of gradient descent. Given the current value of the weight vector Λ, we will update it by a small step in the direction of the steepest increase of the likelihood, given by the vector of partial derivatives:

$$\lambda_f^{k+1} \longleftarrow \lambda_f^k + \beta \frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_f} = \lambda_f^k + \beta \left(\tilde{E}[f'] - \hat{E}[f'] \right) \tag{18}$$

Equation (18) will be applied iteratively, for all $f \in \mathcal{F}$, until the change in likelihood is smaller than some preselected threshold. Note that while $\tilde{E}[f]$ is computed only once for each feature f, we will have to re-compute the value $\hat{E}[f]$ after every update. This makes the learning procedure quite expensive. However, the learning procedure is guaranteed to converge to the global optimum. Convergence is ensured by the fact that the objective function $\mathcal{L}_{\hat{P}}$ is $\cap$-convex with respect to the weights λ_f. One may verify this by computing the second-order derivative of $\mathcal{L}_{\hat{P}}$ and observing that it is everywhere negative.

4.8 Field Induction Algorithm

We are finally ready to bring together the results of the previous subsections into one algorithm for automatic induction of random fields models for polyphonic music:

1. **Initialization**

   (a) Let the feature set $\mathcal{F}^0$ be the set of single-note features: $\mathcal{F}^0 = \{n_{i,t} : i = 0 \dots 11\}$

   (b) Set the initial features weights $\lambda_f = 1$ for all $f \in \mathcal{F}^0$

2. **Weight Update**

   (a) Set $\lambda_f^{k+1} \leftarrow \lambda_f^k + \beta \left(\tilde{E}[f] - \hat{E}[f] \right)$ for each feature $f \in \mathcal{F}$

   (b) If there is noticeable change in likelihood, repeat step (2a)

3. **Feature Induction**

   (a) Enumerate the set of candidate features

   (b) For every candidate feature $g \in \mathcal{G}$ compute the optimal weight $\alpha_g = \log \left[\frac{\tilde{E}[g](1-\hat{E}[g])}{\hat{E}[g](1-\tilde{E}[g])} \right]$

   (c) For every $g \in \mathcal{G}$ compute expected improvement (gain) from adding g to the structure $\mathcal{F}$

   (d) Pick the candidate g that promises the highest improvement, add it to the structure $\mathcal{F}$, and set $\lambda_g = \alpha_g$

   (e) If there is noticeable change in likelihood, go to step (2), otherwise return $\mathcal{F}$ and Λ as the induced field

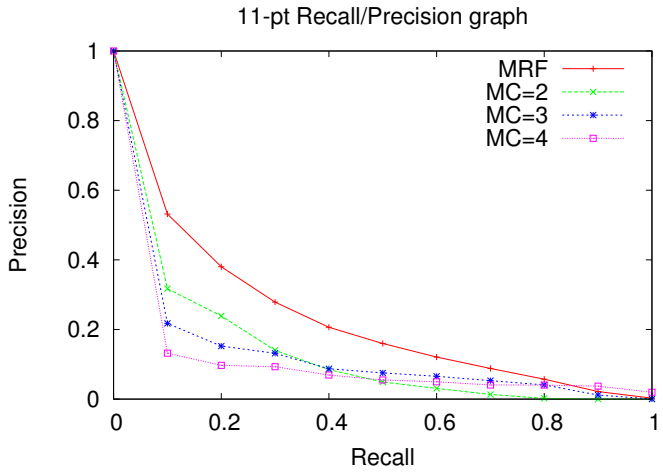
4.9 Discussion on Markov Chains

At this point we should note that our random field approach in some sense encompasses the Markov chain approach. Instead of inducing the features of the field, one could easily preselect as features all possible one-dimensional chains under a certain fixed length and then learn the weights of those features directly. One advantage to our current approach, however, is that by selectively adding only the best features to the model, not only is the final number of parameters much smaller, but the features themselves grow in the direction of the data. In our experiments, for some songs we learned features covering 6 simultaneities (onsets) within the first thousand features induced. A Markov chain would require $12^6 \approx 2.99$ million parameters to cover this same onset range.

4.10 Music Retrieval using Random Fields

Now that we have a framework for creating a Markov Random Field model of a piece of music we wish to use this model for retrieval. We do this by estimating a model from a query and then observing how well that query model predicts the notes in each document in the collection. In other words, our measure of similarity is the expected *cross-entropy* between the empirical distribution $\tilde{P}_D(n|H)$ of the document and the estimate $\hat{P}_Q(n|H)$ produced by the query model. This measure is essentially equation (7) from above, with the document rather than the query as the target distribution.

If the model estimated from the query does well at predicting the notes in a piece of music, then the query and that piece could have been drawn from the same underlying distribution. Therefore we regard them as "similar". This process is repeated for all pieces in the collection, and pieces are then ranked in order of increasing cross-entropy (dissimilarity).



	Θ_{MRF}	$\Theta_{MC=2}$		$\Theta_{MC=3}$		$\Theta_{MC=4}$	
			%Change		%Change		%Change
Retrieved:	151000	151000		151000		151000	
Relevant:	8801	8801		8801		8801	
Rel|ret:	7291	4508	-38.17*	6805	-6.67*	6037	-17.20*
Interpolated Recall - Precision							
at 0.00	1.0000	1.0000	0.0	1.0000	0.0	1.0000	0.0
at 0.10	0.5316	0.3171	-40.3*	0.2176	-59.1*	0.1319	-75.2*
at 0.20	0.3802	0.2388	-37.2*	0.1522	-60.0*	0.0970	-74.5*
at 0.30	0.2787	0.1404	-49.6*	0.1316	-52.8*	0.0931	-66.6*
at 0.40	0.2063	0.0848	-58.9*	0.0873	-57.7*	0.0691	-66.5*
at 0.50	0.1598	0.0492	-69.2*	0.0754	-52.8*	0.0550	-65.6*
at 0.60	0.1209	0.0307	-74.6*	0.0656	-45.7*	0.0498	-58.8*
at 0.70	0.0880	0.0134	-84.8*	0.0530	-39.8*	0.0406	-53.9*
at 0.80	0.0571	0.0021	-96.2*	0.0407	-28.7*	0.0404	-29.2*
at 0.90	0.0218	0.0000	-100.0*	0.0116	-46.8*	0.0372	+70.5*
at 1.00	0.0028	0.0000	-100.0*	0.0002	-91.2*	0.0197	+603.4*
Average precision (non-interpolated) over all rel docs							
	0.2175	0.1245	-42.76*	0.1021	-53.05*	0.0790	-63.68*

Figure 6: Recall-Precision results for Markov Random Field versus Markov Chain models

5 EXPERIMENTS AND ANALYSIS

For evaluation, we have assembled four collections. The first is a set of approximately 3000 polyphonic music pieces from the CCARH at Stanford. These are mostly baroque and classical pieces from Bach, Beethoven, Corelli, Handel, Haydn, Mozart and Vivaldi. Longer scores have sometimes been broken up into their various movements, but otherwise each piece is unique. Our remaining three sets of music are pieces which were intentionally composed as variations on some theme:

Twinkle 26 individual variations on the tune known to English speakers as 'Twinkle, twinkle, little star' (in fact a mixture of mostly polyphonic and a few monophonic versions);

Lachrimae 75 versions of John Dowland's 'Lachrimae Pavan' from different 16th and 17th-century sources, sometimes varying in quality (numbers of 'wrong' notes, omissions and other inaccuracies), and in scoring (for solo lute, keyboard or five-part instrumental ensemble);

Folia 50 variations by four different composers on the well-known baroque tune 'Les Folies d'Espagne'.

For retrieval, we select a piece from the three sets of variations and use that as the query. All other pieces from that same variation set are judged relevant to the query, and the rest of the collection is judged non-relevant. This process is repeated for all pieces in all three sets of variations, for a total of 151 queries.

Let us define Θ_{MC} as the retrieval system based on Markov Chain models, and Θ_{MRF} as the retrieval system based on Markov Random Field models. Figure 6 shows the recall-precision results for the ranked lists produced by each of the various modeling approaches, Θ_{MRF} as well as Θ_{MC} with the length of the chain set to 2, 3, and 4 (1^{st}-, 2^{nd}-, and 3^{rd}-order models, respectively). In the table, Θ_{MRF} is shown first, and each of the Θ_{MC} systems are shown in comparison, with percentage change (whether positive or negative) and an asterisk to indicate statistical significance (t-test at a 0.05 level).

No matter the chain length, the random field approach outperforms the Markov chain approach on just about every level of precision-recall. On average, the Θ_{MC} systems are from 42% to 63% worse. These results show the value of the random field approach.

We believe what is happening is that the random fields are less sensitive to the "noise" that appears with vari-

ations. For example, suppose there is an insertion of a single note in one of the variations. The Markov chain approach counts all possible paths through a polyphonic sequence. The number of these paths is exponential in the length of the chain. A single note insertion therefore disproportionately increases the number (and character) of paths being counted. This analysis is borne out by the fact higher-order Markov chains actually do progressively worse (see Figure 6). The longer the chain, the more a single insertion affects the model. Two note insertions create even more paths.

On the other hand, random fields take into account the dependencies between the features. They work by calculating feature expectations over the data (see section 4.5), and adjusting the λ weights so that the contribution of each feature toward the prediction of a note label ("on" or "off") is balanced. A single note insertion may activate a few additional feature functions, but the contribution of these additional features do not throw off the overall correctness of the note probability estimate because their λ weights are learned initially with non-independence in mind. Random fields are more robust when it comes to detecting variations.

6 CONCLUSION

In this paper we developed a retrieval system based on automatically-induced random fields and show the superiority of these models over Markov chains. Central to our approach is the notion of a binary feature function, a conjunction of notes positioned at fixed pitches and locations.

Yet these features are not the only ones possible. Recall from section 4.2 that features are just functions that return a boolean value. What happens inside the function is as limitless as one need imagine. For example, one can create models of rhythmic patterns by choosing onset conjunction features of the sort: f_1 = "was there an onset 100 ticks prior to the current onset, and another onset 300 ticks prior to the current onset?" (We are currently developing such models.)

One is not limited to rhythm alone, any more than one is limited to pitch alone. Alongside pitch-only and rhythm-only features, our set of active feature functions $\mathcal{F}$ can contain features that are a mixture of both pitch and duration. E.g.: Let f_3 = "the previous note was an E and it lasted for 200 ms". Not only does this feature contain both pitch and duration information, but if the model already contains the pitch-only feature f_2 = "the previous note was an E", we may add f_3 without worry. The training that occurs as part of weight updating (section 4.7) insures that the λ values given to all feature are balanced, to automatically take into account statistical dependencies between features. (It should be clear that f_2 and f_3 are not independent.) Early work by Doraisamy and Rüger (2001) and Lemström et al. (1998) experimented with features of this nature, but were forced by lack of framework to make the independence assumption. With random fields, this assumption is no longer necessary.

Finally, it goes without saying that in addition to pitch and rhythm, as long as one can craft a binary wrapper (feature function) one can use any other type of musical information one has available, including metadata and data obtained from semantic analysis of audio. Features could include functions such as: f_4 = "my timbre classifier gives me confidence > 0.9 that there is a trombone playing at this point in this audio recording, and my beat tracker estimates the current tempo for this song at between 80 and 100 bpm, and there was an onset around 300 ticks ago, and the metadata tells me this song was recorded in the 1970s" This feature function may (when properly weighted and combined with the rest of $f \in \mathcal{F}$) be a strong positive indication that the note C# is "on".

Random fields are a framework for sequential music modeling in which combination of multiple, non-independent sources and types of data may be explored. Markov random fields are more robust than Markov chains. They are accurate, without overfitting, as we can see from the retrieval results above. They also offer a method for attaching relative importance to various features without having to make independence assumptions between the features used. In short, they are an important framework for developing the kinds of models needed for music information retrieval applications.

7 ACKNOWLEDGEMENTS

Many thanks to Victor Lavrenko, whose early collaboration made retrieval systems of this sort possible. We also wish to thank the conference reviewers for their invaluable comments.

REFERENCES

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. In *Ann. Math. Statistics, 43*, pages 1470–1480, 1972.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, pages 380–393, 1997.

S. Doraisamy and S. M. Rüger. An approach toward a polyphonic music retrieval system. In J. S. Downie and D. Bainbridge, editors, *Proceedings of the 2nd Annual ISMIR*, pages 187–193, Indiana University, Bloomington, Indiana, October 2001.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

K. Lemström, A. Haapaniemi, and E. Ukkonen. Retrieving music - to index or not to index. In *The Sixth ACM International Multimedia Conference (MM '98)*, pages 64–65, Bristol, UK, September 13-16 1998.

C. Zhai and J. Lafferty. Dual role of smoothing in the language modeling approach. URL `citeseer.ist.psu.edu/zhai01dual.html`.