Polyphonic Music Modeling with Random Fields

Victor Lavrenko, Jeremy Pickens
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts, Amherst, MA 01003 USA
{lavrenko,jeremy}@cs.umass.edu

ABSTRACT

Recent interest in the area of music information retrieval and related technologies is exploding. However, very few of the existing techniques take advantage of recent developments in statistical modeling. In this paper we discuss an application of Random Fields to the problem of creating accurate yet flexible statistical models of polyphonic 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. We offer an evaluation of these models in terms of perplexity and prediction accuracy, and show that random fields not only outperform Markov chains, but are much more robust in terms of overfitting.[1]

Categories and Subject Descriptors

H.5.5 [**Sound and Music Computing**]: Modeling

General Terms

Algorithms

1. INTRODUCTION

In this paper we discuss an application of Random Fields to the problem of statistical modeling of polyphonic music. Statistical models of music are relatively few, despite the fact that music is a fairly well-understood phenomenon, familiar to everyone since earliest years and intimately tied to the human experience. This is partly due to the fact that the focus of contemporary musical theory is primarily in the area of grammar and semantics, and there have been little need or interest in modeling music as a stochastic process. However, as music gradually finds its home in a digital world, there will come a growing need for systems that facilitate organizing, searching and classifying musical collections, systems that can verify authenticity or determine authorship of a given musical piece, or perhaps even mix, blend and re-style compositions to the user's desire. These systems are likely to be based on statistical models of music, just like the modern text-processing systems are largely based on the statistical models of language.

In particular, we suggest three specific applications which may be aided by our statistical modeling process. First, ad hoc music retrieval is made possible in that a model may be created for every piece of music in the collection, then each piece ranked by the probability that model produces a given query. Second, audio music recognition (AMR), the transcription of raw audio into symbolic notation [13, 2], is a difficult task for polyphonic music, especially when note candidates are only considered in their local context. The models we create in this paper may be used to better inform local decisions based on patterns detected throughout a piece as a whole. In this manner, imperfect transcriptions may be cleaned up, missing notes inserted, extraneous notes deleted, by the statistical patterns inducted by our models. Finally, optical music recognition (OMR), the transcription of scanned image sheet music into symbolic notation [10], may be aided by our approach in a manner similar to the AMR application as well.

As we will describe in the following sections, music can be thought of as a very structured two-dimensional stochastic process. Unlike text, the musical vocabulary is relatively small, containing at most several hundred discrete note symbols. What makes music so fascinating and expressive is the very rich structure inherent in musical pieces. Whereas text samples can be reasonably modeled using a simple bi-gram or tri-gram language model, musical samples are characterized by numerous symmetries, repetitions, and short- and long-term interactions that are beyond the capabilities of simple Markov chains. This is one of the main reasons why we selected Random Fields as the paradigm for modeling music.

Random Fields represent a generalization of Markov chains to multi-dimensional spatial processes. They are incredibly flexible, allowing us to model arbitrary interactions between the notes in a sample of music. They have been very popular in modeling of physical systems, and recently have demonstrated superior performance in a number of text-related applications. 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.

[1] This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-9905842, and in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

MM'03, November 2–8, 2003, Berkeley, California, USA.

The remainder of this report is organized as follows: In section 2 we provide a brief overview of related work, both in music modeling and in recent applications of random fields. Section 3 starts with a discussion of polyphonic music and introduces the music representation that will be used throughout this report. In section 4 we discuss how a sample of music can be mapped onto a two-dimensional field over binary variables. We discuss the structure of the field, the interactions between the variables and the procedures that can be used to induce the field and learn its parameters. In section 5 we discuss the performance of our system and provide a quantitative evaluation of its effectiveness. Section 6 serves to summarize the findings of this project.

2. RELATED WORK

This section serves a dual purpose. First we will briefly survey recent publications attempting to model music as a stochastic process. We will also discuss the fundamental differences between these approaches and the model we explore in the paper. Then we will discuss relevant literature on applications of random fields in the area of natural language processing.

2.1 Music Models

Due to the complex nature of polyphonic music, most of the work in stochastic modeling of music, as in the music information retrieval field in general, operates in the monophonic domain. Examples and applications include automated score following [7] and melody-based information retrieval [1]. Notable exceptions to the monophonic domain are Raphael [17] and Pickens et al [16]. The former work uses carefully constructed hidden Markov models to automatically transcribe polyphonic audio piano music, treating the actual notes (pitch values) as the hidden states and using Viterbi to discover these values. The latter work proceeds by mapping each simultaneity in a piece of music onto a set of 24 triads using a musically-motivated heuristic algorithm, then constructing “visible” n-gram Markov models of those triad sequences.

Finally, it must be mentioned that entropy has been applied to other areas of music, including clustering and melodic extraction. As an example of the latter, Uitdenbogerd [19, 20] uses entropy to select monophonic melody lines from polyphonic MIDI pieces. What we present in this paper is a more comprehensive application of the principles of entropy to the problem of creating reasonable music models, which models may then be used for a wide variety of music-related tasks.

2.2 Random Fields and Maximum Entropy

Markov Random Fields have long been a popular tool for modeling complex physical systems, and most of their fundamental properties were derived in a physical setting. The technique has also been quite popular in the field of computer vision. However, it is only recently that random fields found applications in large-vocabulary applications, such as language modeling and information extraction. One of the most influential works in the area is the 1997 publication of Della Pietra et al. [6], which outlined the algorithms that will be used in parts of this paper. The learning procedures for parameter estimation date back to a 1972 publication by Darroch and Ratcliff [5], but do not include the techniques for feature induction, which are crucial for the purpose of music modeling. Berger et al. [3] were the first to suggest the use of maximum entropy models for natural language processing. Since then, the models have steadily gained popularity, and variations were proposed for the tasks of language modeling [18], text segmentation and information extraction [12, 9], text classification [14] and machine translation [15]. Recently, Malouf [11] carried out an extensive comparison of learning algorithms for the maximum-entropy framework.

While our work was inspired by applications of random fields to language processing, it bears more similarity to the use of the framework by the researchers in computer vision. In most natural language applications authors start with a reasonable set of features (which are usually single words, or hand-crafted expressions), and the main challenge is to optimize the weights corresponding to these features. This works well in natural language, where words bear significant semantic content. In our case, induction of the random field is the crucial step. As we describe below, we will start with a set of single notes, which by themselves cannot reflect anything about a musical piece. We will use the techniques suggested by [6] to automatically induce new high-level features, such as consonant and dissonant chords, progressions and repetitions.

3. MUSIC REPRESENTATION

The domain in which we wish to apply the MRF modeling is polyphonic music. 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. Music in this form contains all the notes in a piece of music, the onset, symbolic-code duration (eg: "quarter" note, "half" note"), and pitch of every single note. This music also comes complete with time signatures, key signatures, sharps, flats, ties, slurs, and various other dynamics markings which help instruct the performer as to the manner in which the piece should be performed. We deal with neither of these forms, but rather with a form that combines aspects of the two. MIDI (www.midi.org) is a characteristic *example*, though we do not require the music to be in this specific format. In MIDI files, the onset, duration, and pitch of every note in a piece of music is known. But no other information is necessarily available. The pitches are encoded as numbers, ranging from 1 to 128. The durations are not symbolic, but instead are given as millisecond integers. The onset times also are not symbolic, but occur at millisecond integer locations.

Monophonic music is such that, if a note is playing, no new note may start until the previous note has finished. In polyphonic music, there is no such restriction. Any note may start or finish before any other note finishes. We may therefore think of polyphonic MIDI music as a two-dimensional graph, with millisecond 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 represent notes being "on". White circles represent notes being "off".

MIDI Note #60	●	●	●	○	○	○	○	●	●	●	●
MIDI Note #61	○	○	○	○	●	●	○	○	○	●	●
MIDI Note #62	○	●	●	●	●	●	○	●	●	○	○

In order to do our MRF modeling, we need to select features from our polyphonic source documents and use those features for modeling. We begin by selecting only the onset times of each new pitch in the sequence, and ignoring the duration of the note. The example above thus transforms into:

MIDI Note #60	●	○	○	○	○	○	○	●	○	○	○
MIDI Note #61	○	○	○	○	●	○	○	○	○	●	○
MIDI Note #62	○	●	○	○	○	○	○	●	○	○	○

Next, we get rid of all millisecond onset times which contain no pitches. (We are throwing away not only the duration of the notes, but the duration between notes. We feel this is necessary for a first-stage modeling attempt. Future models might contain more complexity.) Those millisecond onset times which do contain pitches, however, we give the specialized name "simultaneity".

MIDI Note #60	●	○	○	●	○
MIDI Note #61	○	○	●	○	●
MIDI Note #62	○	●	○	●	○

Finally, we reduce the 128-note y-axis to a 12-note octave-equivalent pitch set. We do this simply by taking the mod-12 value of every MIDI pitch number. The example above thus becomes:

Note #0	●	○	○	●	○
Note #1	○	○	●	○	●
Note #2	○	●	○	●	○

So we are left with a sequence of 12-element bit vectors; there is either a 1 or a 0. In each spot, depending on whether a note of that (mod 12) pitch had an onset in that particular simultaneity.

As an important aside, we must again emphasize that the modeling techniques described in this paper do not require the source music to be MIDI. The only information with which we are concerned is the pitch values and relative order of the notes. The section above explains how we might obtain this information from a MIDI file, but we could have just as easily obtained it from a transcribed (AMR) audio file or a transcribed (OMR) scanned sheet music file, to name just a few sources. Our algorithms work on any music data in which pitch information is available.

4. RANDOM FIELDS OVER MUSIC

In the previous section we showed how polyphonic music can be represented as a temporal progression of 12-dimensional binary vectors. Restricting the representation to a single octave (12 notes) is more of an implementation convenience, and in general we can think of music as a sequence of n-dimensional binary vectors. Given this representation, it is tempting to model music with simple Markov Chains, treating the entire vectors as if they were words in some vocabulary, and computing the probabilities of transition from one vector to another. This is a viable approach, but it suffers from extreme data sparseness. Instead, we choose to treat the entire progression as a two-dimensional field over binary variables. This framework allows us to selectively model the interactions between the individual notes. For instance, with random fields we could directly model the probability of the same note being played twice in a row. Modeling this event in isolation is next to impossible with Markov Chains on entire 12-bit vectors.

In the remainder of this section we will discuss in detail our model for polyphonic music. First, we will discuss the structure of a random field over the notes, the notions of history or neighborhood and feature functions over the neighborhood. We will also give the general form of the maximum entropy distribution over the variables in the field, and state the objective of our model. Then, in section 4.2 we will describe how a suitable field structure can be induced automatically, starting with primitive atomic features and generalizing to more interesting complex relationships. The field induction procedure closely follows the algorithm described in [6], the primary difference being that we are dealing with a directed conditional field, whereas [6] worked with an undirected joint model. Finally, in section 4.3 we discuss how we can learn the parameters of the model to optimize its performance on the chosen objective function.

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

Figure 1: Variables of a Musical Random Field

4.1 Structure 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 $t = 1 \dots T$, and by note index $i = 0 \dots 11$. Each variable $\{n_{i,t}\}$ can be either 0 or 1, indicating whether a note i is on or off at time t. Figure 1 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 variables, or restrict the conditioning to the immediate neighbors of $n_{i,t}$. On the contrary, we believe that 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.

4.1.1 Directed Structure

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 $\{n_{j,s>t}\}$. In our initial 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:

$$H_{i,t} = \{n_{j,s} : s < t\} \cup \{n_{j,s} : s = t, j < i\} \qquad (1)$$

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. By defining the history in this manner we have converted our model from a general Markov Random Field to a directed structure, where all influences progress in an acyclic fashion left-to-right. We would like to stress that in our opinion this does not make the model equivalent to a simple Markov Chain, since we still allow arbitrary dependencies within the subset defined by the neighborhood $H_{i,t}$. For a graphical illustration of directed dependencies in the field see Figure 2.

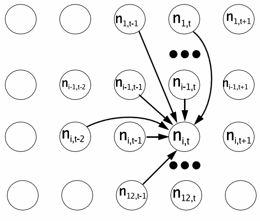

Figure 2: Dependencies in a directed random field over polyphonic music: note $n_{i,t}$ is influenced by the neighborhood $H_{i,t}$ of notes preceding it.

4.1.2 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 dependencies to binary questions of the form: "*was note j played at some 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 $n_{j,s}{\in}S$:

$$f_S(n_{i,t}, H_{i,t}) = n_{i,t} \prod_{n_{j,s} \in S} n_{j,s} \tag{2}$$

Defined in this manner, our feature functions are always binary, and equal to 1 if all the notes defined by S were played before the target note $n_{i,t}$. The binary nature of our features will come particularly handy when we describe the details of automatic field induction in the following section. Also note that 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.

On a final note, 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 $n_{i,t}$. 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, below we discuss some examples of features that could have an impact on note "2" at time t. Figure 3 shows a graphical representation of the same notes:

1: $f_1(n_{2,t}, H_{2,t}) = n_{2,t}n_{1,t}$ – this feature represents the event that two notes immediately adjacent on the musical scale (notes 1 and 2) were being played at once. This is a highly unlikely event in a typical musical piece, so we expect this feature to have a negative contribution to the probability that $n_{i,t} = 1$.

2: $f_2(n_{2,t}, H_{2,t}) = n_{2,t}n_{2,t-1}n_{2,t-2}$ – this feature represents the event that the same note (2) was played three times in a row. This is a relatively common event, and we expect this feature to possess a positive weight.

3: $f_3(n_{2,t}, H_{2,t}) = n_{2,t}n_{1,t}n_{3,t-1}n_{3,t-2}$ – hypothetical feature, representing both a dissonant combination (notes 1 and 2 at the same time), and a reasonably common progression (note 3 twice, and then note 2).

4: $f_4(n_{2,t}, H_{2,t}) = n_{2,t}n_{0,t}n_{2,t-2}n_{0,t-2}$ – complex feature representing a chord (notes 0 and 2) played successively, but with some other intervening set of notes which do not participate in the formulation of the given feature.

4.1.3 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:

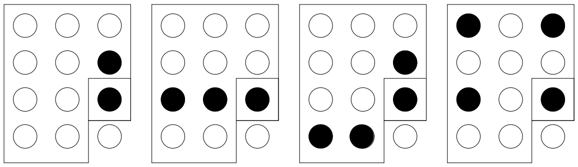

Figure 3: Examples of musical features that may be induced to predict the probability of note 2 being played at time t. Black circles represent notes that are part of the feature function. Boxed black circle denotes the note $n_{2,t}$. Boxed area represents the history $H_{2,t}$. From left to right,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

$$\hat{P}(n_{i,t}|H_{i,t}) = \frac{1}{Z_{i,t}} \exp\left\{ \sum_{f\in\mathcal{F}} \lambda_f f(n_{i,t}, H_{i,t}) \right\} \tag{3}$$

In equation (3), 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. $Z_{i,t,\Lambda,\mathcal{F}}$ 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:

$$Z_{i,t} = \sum_n \exp\left\{ \sum_{f\in\mathcal{F}} \lambda_f f(n, H_{i,t}) \right\} \tag{4}$$

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 (4) is not feasible. In our case, the special nature of the underlying problem makes computation of the partition function extremely simple. Recall that all underlying field variables are binary, so equation (4) 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 (3) and (4) is zero whenever $n_{i,t} = 0$. These observations allow us to re-write equation (3) in a form that allows very rapid calculations:

$$\begin{aligned} \hat{P}(n_{i,t} = 1|H_{i,t}) &= \sigma\left\{ \sum_{f\in\mathcal{F}} \lambda_f f(1, H_{i,t}) \right\} \\ \hat{P}(n_{i,t} = 0|H_{i,t}) &= 1 - P(n_{i,t} = 1|H_{i,t}) \end{aligned} \tag{5}$$

Here σ is the sigmoid function, defined as: $\sigma(x) = \frac{e^x}{1+e^x}$. We have stated equation (5) as a special case applicable to our particular setting. In the remaining arguments we will use the form given by equations (3) and (4) to ensure generality.

4.1.4 Objectives

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 $\mathcal{T}$:

$$\begin{aligned} \mathcal{L}_{\hat{P}} &= \frac{1}{12T} \log \prod_{t=1}^{T} \prod_{i=0}^{11} \hat{P}(n_{i,t}|H_{i,t}) \\ &= \sum_{H} \sum_{n} \tilde{P}(n,H) \log \hat{P}(n|H) \end{aligned} \tag{6}$$

In the second step in equation (6) 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:

$$\tilde{P}(n,H) = \frac{1}{12T} \sum_{t=1}^{T} \sum_{i=0}^{11} \delta(n, n_{i,t}) \delta(H, H_{i,t}) \tag{7}$$

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

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Returning our attention to equation (6), 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 (6), whereas the second part (bottom) comes very handy in the algebraic derivations of the field induction algorithm.

To summarize, in the previous two sections 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 $\mathcal{T}$:

$$\begin{aligned} \tilde{E}[f] &= \sum_{H} \sum_{n} \tilde{P}(n,H) f(n,H) \\ &= \frac{1}{12T} \sum_{t=1}^{T} \sum_{i=0}^{11} f(n_{i,t}, H_{i,t}) \end{aligned} \tag{9}$$

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:

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

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 (9) and (10) 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:

$$H_{\hat{P}} = \sum_{H} \tilde{P}(H) \sum_{n} \hat{P}(n|H) \log \hat{P}(n|H) \tag{11}$$

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 (6). Della Pietra et al. [6] provide an excellent proof of this claim.

4.2 Feature Induction

In the previous section we outlined the general structure of a directed 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 (6)). Recall that we selected exponential form for $\hat{P}(n|H)$. If we examine equation (3) 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 can 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 the 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 and slowly improving on that. Then, in section 4.3 we describe how a set of weights Λ can be optimized for a given structure $\mathcal{F}$.

Our approach to inducing the structure of the field closely follows the algorithm proposed by Della Pietra et al. [6]. We start with a very flat structure of the field that contains only individual notes, without any dependencies: $\mathcal{F}^0 = \{n_{i,t} : i = 0 \ldots 11\}$. Recall that our features are time-invariant, so t is a dummy index that only serves as a point of reference in the field. 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 $\mathcal{F}$:

$$\mathcal{G} = \left\{ \begin{array}{lcl} f_S \cdot n_{j,s} & : & f_S \in \mathcal{F} \text{ and there exists} \\ & & n_{j',s'} {\in} S \text{ such that } |s-s'| \leq 2 \end{array} \right\} \tag{12}$$

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 [6], 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:

$$\mathcal{L}_{\hat{P}+\{\alpha g\}} = \mathcal{L}_{\hat{P}} + \alpha \tilde{E}[g] - \log \hat{E}[e^{\alpha g}] \tag{13}$$

We may factor this expression in terms of three convenient components: the old log-likelihood $\mathcal{L}_{\hat{P}}$ which does not depend on g or α, the empirical expectation $\tilde{E}[g]$, and the predicted expectation $\hat{E}[e^{\alpha g}]$. If the candidate feature g is binary, that is $g(n,H) \in \{0,1\}$, the new log-likelihood $\mathcal{L}_{\hat{P}+\{\alpha g\}}$ can be expressed in a particularly convenient form that involves only expectations over g:

$$\mathcal{L}_{\hat{P}+\{\alpha g\}} = \mathcal{L}_{\hat{P}} + \alpha \tilde{E}[g] - \log \left[e^{\alpha} \hat{E}[g] + (1 - \hat{E}[g]) \right] \tag{14}$$

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 find the root of the derivative:

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

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:

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

Equation (16) is the result of substituting the optimal value of α from equation (15) and some simple algebraic manipulation. 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.

4.3 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. First, we will factor the likelihood into two parts, corresponding to the numerator and the denominator of equation (3):

$$\begin{aligned} \mathcal{L}_{\hat{P}} \;=\; & \sum_H \sum_n \left[\tilde{P}(n,H) \sum_f \lambda_f f(n,H) \right] - \\ & \sum_H \left[\tilde{P}(H) \log \sum_{n'} e^{\Sigma_f \lambda_f f(n',H)} \right] \end{aligned} \tag{17}$$

Now, to maximize the objective, we will, as before compute the partial derivatives of $\mathcal{L}_{\hat{P}}$ with respect to each weight $\lambda_{f'}$, with the intention of driving these derivatives to zero:

$$\frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_{f'}} = \tilde{E}[f'] - \hat{E}[f'] \tag{18}$$

Driving each partial derivative to zero will maximize the objective. But note how this also forces the satisfaction of constraints that $\tilde{E}[f] = \hat{E}[f]$ which we discussed in section 4.1.4 as part of the maximum-entropy framework.

Unfortunately, there is no closed-form solution that would allow us to set the weights to their optimal values. Instead, we will describe an iterative procedure that will drive the weights towards the optimum. There are a number of algorithms for adjusting the weights in the exponential models. The most widely known is perhaps the Generalized Iterative Scaling (GIS) algorithm proposed by Darroch and Ratcliff [5], and the Improved Iterative Scaling version (IIS) described by Della Pietra et al. [6]. However, as Malouf pointed out in his recent analysis [11], iterative scaling is an extremely slow procedure, hindered by the need to compute complex upper bounds for the objective. Much faster convergence can be achieved by using variations of gradient descent. This is the approach we adopt in this paper. 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. We compute the following for all $f \in \mathcal{F}$:

$$\lambda_f^{k+1} \longleftarrow \lambda_f^k + \beta \frac{\partial \mathcal{L}_{\hat{P}}}{\partial \lambda_f} = \lambda_f^k + \beta \left(\tilde{E}[f'] - \hat{E}[f'] \right) \tag{19}$$

Equation (19) will be applied iteratively, until the change in likelihood is smaller than some pre-selected value. 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, we can claim that 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.

4.4 Field Induction Algorithm

We are finally ready to bring together the results of sections 4.1, 4.2, and 4.3 into one algorithm for automatic induction of directed random fields over polyphonic music. Let $\mathcal{T}$ be the training set of T simultaneities of 12 notes each. The field induction algorithm is as follows:

1. **Initialization**

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

   (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 (2.a).

3. **Feature Induction**

   (a) Compute the set of candidate features:
   $\mathcal{G} = \{f_S \cdot n_{j,s} : f_S \in \mathcal{F}$ and $\exists n_{j',s'} \in S$ s.t. $|s - s'| \leq 2\}$

   (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 from adding g to the structure $\mathcal{F}$:

   $$\mathcal{L}_{\hat{P}+\{\alpha_g g\}} - \mathcal{L}_{\hat{P}} = \alpha_g \tilde{E}[g] - \log \left[e^{\alpha_g} \hat{E}[g] + (1 - \hat{E}[g]) \right]$$

   (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.

5. EXPERIMENTS

In this section we briefly report on the experiments we carried out to test the effectiveness of the proposed model in capturing the regularities of polyphonic music. For our project, we have four collections. The first is a set of approximately 3000 polyphonic music pieces from the CCARH [8] at Stanford. These are mostly baroque and classical pieces from Bach, Beethoven, Corelli, Handel, Haydn, Mozart and Vivaldi. The original scores have sometimes been broken up into their various movements, but otherwise each piece in the collection is a unique composition. Our remaining three sets of music, on the other hand, are pieces which were intentionally composed as variations on some original theme.

T 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);

L 75 versions of John Dowland’s ‘Lachrimae Pavan’, collected as part of the ECOLM project (www.ecolm.org) from different 16th and 17th-century sources, sometimes varying in quality (numbers of ‘wrong’ notes, omissions and other inaccuracies), in scoring (for solo lute, keyboard or five-part instrumental ensemble), in sectional structure and in key;

F 50 variations by four different composers on the well-known baroque tune ‘Les Folies d’Espagne’.

For the 3000-piece CCARH data, we split the collection into 1% training and 99% testing subsets. For each of the TLF sets, we split them in to 50%-50% training and testing subsets. We induce a random field as we described in section (4.4) for each of the training sets. For comparison, we also train up a Markov chain model for each of these same training sets.

We will perform two types of evaluation. First, we qualitatively look at the types of features that were induced in our field. Our hope here is that random fields can pick out very intuitive features, something that a musician would easily recognize. Second, we will carry out a brief comparison between our model and the simple (though comparable) Markov Chain model. We will look at which model was able to perform better on the testing data, as a function of the number of parameters in the model.

5.1 Qualitative Evaluation

We would like to see whether the features which are induced and highly weighted by our algorithm are indicative of the same patterns and structures a musician might notice. We must emphasize that our goal in this work is to create a statistical model that accurately predicts which notes will or should appear in a piece of music. We are not trying to do a formal music-theoretic analysis. However, we want to qualitatively show that those features learned by our statistical model do actually make sense musically. There is no reason that they must; it is just interesting that they do. Toward this end we trained a model exclusively on Variation 6 of Mozart’s “Ah vous dirai-je, Maman”, one of the “Twinkle” variations. Our algorithm quickly induced approximately 8,000 features and learned their associated weights. Those induced features with the highest weights are given below.

(We need to make a quick note of the following: For this qualitative evaluation, for the sake of readability, we did not do an octave equivalance mod 12 on the pitch values, as explained in section 3. Instead, we use the middle four octaves centered around middle C. Numbers are given in standard MIDI values, where **60** represents middle C. Notes sounding at the same time are given in brackets, such that [**60**][**79,64**] can be interpreted as note **60** followed by the simultaneous playing of notes **79** and **64**.)

- [**74,67**] Perfect 5th on G (G is the dominant of C)
- [**53**][**52**][**53**] Sequential minor 2nds (as you would find in the 3rd-to-4th degrees of the C Major scale)
- [**81,72,53**] F Major triad (F is the sub-dominant of C)
- [**60**][**79,64**] Semi-arpeggiated C Major triad (first the c, then the e and the g)
- [**72,67,48**] Perfect 5th on C
- [**59**][**60**] Sequential minor 2nd (as you would find in the 7th-to-8th degrees of the C Major scale)
- [**64**][**60**] Sequential Major 3^{rd} on C
- [**79,76,48**] C Major triad
- [**48**][**47**][**48**] Sequential minor 2nds (as you would find in the 7th-to-8th degrees of the C Major scale)

We can see that these highly-weighted features are indeed indicative of the same sort of patterns a musician might notice. First, there are a lot of Major triads, perfect 5^{ths}, and Major 3^{rds}. These chords are also either based on C Major, or on a closely related key to C (F and G). Indeed, this piece is written in the key of C Major. Second, the trill-like arpeggiations on the notes 52-53 and 47-48 are also highly characteristic of this piece. In short, this algorithm has not picked out just any random polyphonic subset of notes; it has managed to select and weight highly features which are characteristic of the basic tonality and structure of this piece of music as a whole.

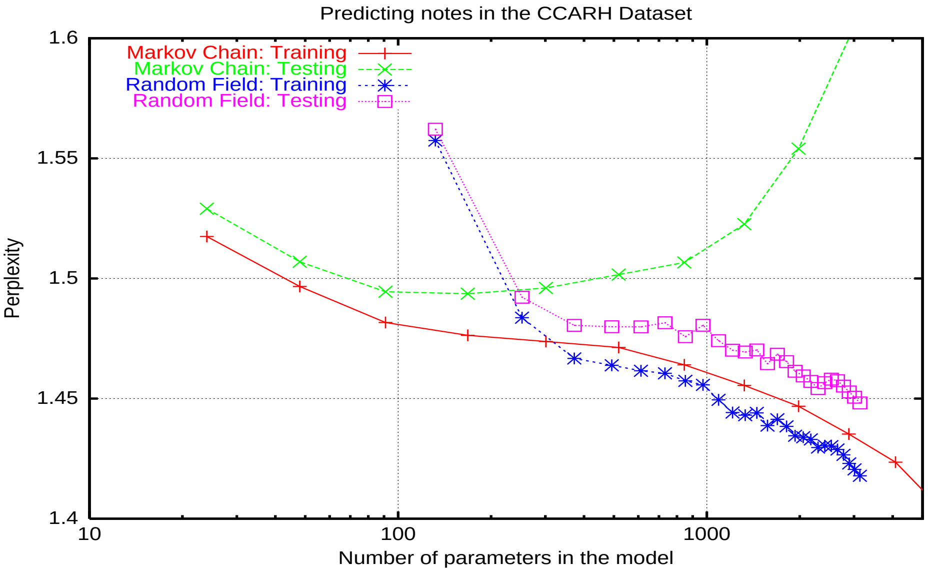

Figure 4: Training and testing perplexity of Markov Chains and Random Fields on the CCARH dataset. Markov Chains show very poor generalization: training perplexity continues to decrease while perplexity of the testing set explodes for $N \geq 6$. Random Fields show very good generalization: testing perplexity trails the training perplexity. Similar results were observed on the other three datasets.

5.2 Quantitative Evaluation

In this section we conduct a quantitative evaluation of using Random Fields for the purpose of polyphonic music modeling. Our main goal is to show that Random Fields are able to learn generative models of music – models that generalize beyond the training set and accurately predict the notes in unseen variations.

5.2.1 Baseline

We will compare effectiveness of Random Fields against the popular *N-gram* Markov Chains. A Markov Chain estimates the probability of observing a 1 in a particular position i, t by conditioning the variable $n_{i,t}$ on *all* preceding variables within a limited horizon. Markov Chains work on sequential data, and we linearize the polyphonic representation using the following mapping: $n_{i,t} \longrightarrow n'_k$ where $k = i + 12t$. Markov Chains do not involve any feature induction, the probability of observing a 1 in a given position k under an N-gram Markov Chain is given by:

$$P_N(n'_k | n'_{k-N+1} \dots n'_{k-1}) = \frac{\#(n'_{k-N+1} \dots n'_k)}{\#(n'_{k-N+1} \dots n'_{k-1})} \quad (20)$$

To make Markov Chains comparable to our Random Field models, we estimate 12 separate Markov Chains. Each of the 12 chains is responsible for predicting notes of a single pitch i, but the probabilities can be conditioned on any other pitch value. It is a well-known fact that Markov Chains are extremely effective in capturing the regularities present in the training data. However, for large values of N these models generalize poorly, as it becomes increasingly unlikely that any conditioning history $n'_{k-N+1} \dots n'_{k-1}$ gets an exact match on the testing data. To alleviate this problem, researchers commonly use *smoothing* techniques, where the high-order probability is interpolated with lower-order estimates. We use linear interpolation with the following recursive definition:

$$\hat{P}_N(n'_k | H'_k) = \lambda_N P_N(n'_k | H'_k) + (1 - \lambda_N)\hat{P}_{N-1}(n'_k | H'_k)$$

We set smoothing parameter λ_N according to the estimate originally proposed by Witten and Bell [21]:

$$\lambda_N = A_N / (A_N + V_N)$$

Here A_N is the *aggregate* count of all N-grams that occur in the training data and U_N is the number of *unique* training N-grams. The intuition behind the formula is to provide more smoothing when the training data is sparse and contains many unique N-grams (large U_N) with few repetitions (small A_N).

5.2.2 Evaluation Methodology

As explained at the beginning of section 5, we randomly split each of our collections of music into training and testing sets. We use the training portion to estimate the model and then measure how well our model predicts each note in the testing portion. We use two popular measures of performance to assess predictive capabilities of our model: *perplexity*, commonly used to evaluate generative models, and *prediction accuracy*, favored by the machine learning community.

Perplexity is a measure of probability mass that would be assigned to the correct notes if we attempted to "generate" the testing set by random sampling from our statistical model. Perplexity is defined as the inverse geometric average of probabilities for every note in the testing set: $\left[\prod_{t=1}^{N} \prod_{i=1}^{12} \hat{P}(n_{i,t} | H_{i,t})\right]^{-1/12T}$ Here the products go over every note in every simultaneity of the testing set. From the definition, it is evident that perplexity is just 2 raised to the entropy. Perplexity has an intuitive interpretation in terms of uncertainty about the predicted value: a perplexity of k is the same level of uncertainty one would experience when confronted with k equally-likely choices.

While perplexity is a natural measure for evaluating generative models, a number of researchers [4] observed that perplexity does not always correspond to prediction accuracy. In addition to perplexity values we show *Receiver Operating Characteristic (ROC)* curves which directly show misclassification rates. An intuition behind ROC curves is as follows. We would like to predict

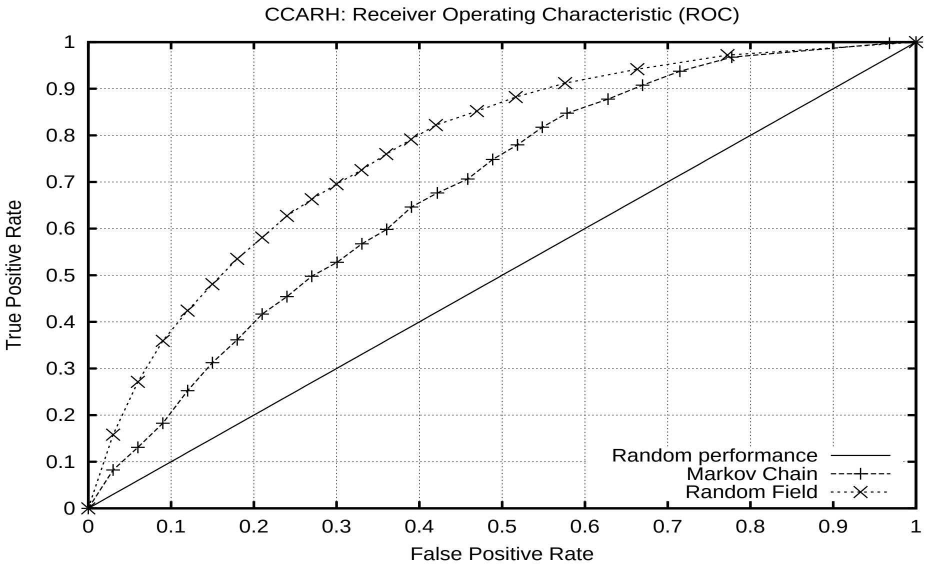

Figure 5: Performance of Random Fields and Markov Chains on the task of predicting notes in the testing portion of CCARH. Random Fields are substantially more accurate than Markov Chains. Similar results were observed on the other three datasets.

the value of a particular note $n_{i,t}$. One way to do that is to look at the probability $\hat{P}(n_{i,t}|H_{i,t})$ assigned by our model and decide $n_{i,t} = 1$ if that probability is above some threshold θ. For any particular threshold we can compute the $true\ positive\ rate$: proportion of all notes $n_{i,t} = 1$ that scored above that threshold, and $false\ positive\ rate$: proportion of all notes $n_{i,t} = 0$ that scored above the threshold. An ROC curve is obtained by plotting the true positive rate against the false positive rate across all possible values of a threshold $\theta \in [0, 1]$.

5.2.3 Perplexity Results

Figure 4 shows the perplexity of the two models on the training and testing portions of the CCARH polyphonic collection. We plot perplexity versus the total number of parameters in the model – for Markov Chains this is the total number of N-grams that have been memorized, for Random fields it is the total number of features we induced. Every point on the Markov Chain curve corresponds to a different "order": the first point reflects a unigram model, the second, a bigram, and so on. The last point in each curve corresponds to a 30-gram Markov Chain. For Random Field curves every point corresponds to 10 iterations of the learning algorithm. Several things become apparent when we examine Figure 4 in detail. First, we see that Markov Chains are effective in capturing the training data: training perplexity drops steadily and reaches very low values. Remember that perplexity near 1 means that the model is almost certain (on average) about the value of each variable. However, we also observe that Markov Chains generalize poorly to testing data. The testing perplexity starts to rise afer we condition on 3 or 4 variables in the past, and beyond 7- or 8-gram predictions are completely off target. After conditioning on 12-15 variables the testing perplexity is around 2, which means that the model is almost clueless about the value of each testing note. This kind of behavior is understandable: high-order Markov Chains learn by $memorizing$ portions of the data, and long memorized histories are unlikely to re-appear in new music.

Random Fields show a very different behavior as we induce more and more features. The training perplexity of these models is not always as low as with N-gram models, but the testing perplexity does not explode and in most cases continues to decrease as we add more and more parameters. This indicates that the model is extracting salient features of the training dataset, instead of memorizing portions of it. These salient features generalize well to the testing portion of the dataset. The lowest testing perplexity achieved by Random Fields is lower than the lowest Markov Chain perplexity. We observed similar results on the other three datasets, see Table 1 for details.

5.2.4 Prediction Accuracy

Figure 5 compares the accuracy of Random Fields and Markov Chains on the task of predicting every note in the testing portion of CCARH. The diagonal line in the plot represents prediction accuracy that would correspond to making random guesses about each note. We observe that Random Fields noticeably outperform Markov Chains. The lower portion of Table 1 summarizes the quantitative difference between the ROC curves on the four datasets. We use area under the ROC curve as a single-number measure of relative performance. The Random Field Model is 13% more accurate than Markov Chains on the CCARH dataset and 11% more accurate on the Lachrimae dataset. It is also important that the Random Field ROC curve completely dominates the Markov Chain ROC curve: this means that the improvement is consistent across all possible threshold values.

6. CONCLUSIONS

In this report we described how Random Fields can be used to model polyphonic music. The model we proposed is theoretically sound, and does not involve any heuristic steps. We directly modeled occurrences of the notes by automatically inducing complex dependencies between sets of notes played at different times in the musical piece. We described the structure of the model, and provided a detailed account of how the model may be trained. We demonstrated that a model is capable of finding very interesting interactions and dependencies within the music. We also showed that our model outperforms Markov Chains in four different collections.

Dataset	CCARH	Twinkle	Lachrimae	Folia
Number of variations	2,806	26	75	50
Total number of notes	15,904,356	15,108	436,152	63,612
Internal consistency (Scale 1 to 10)	Very low (1)	Low (5)	Low (6)	Medium (7)
Min. Perplexity: Markov Chain	1.494 (4-gram)	1.345 (2-gram)	1.375 (3-gram)	1.320 (3-gram)
Min. Perplexity: Random Field	1.448 (260 it.)	1.340 (140 it.)	1.319 (250 it.)	1.311 (750 it.)
Area under ROC: Markov Chain	0.700	0.745	0.776	0.828
Area under ROC: Random Field	0.791 (+13%)	0.811 (+9%)	0.864 (+11%)	0.857 (+3%)

Table 1: Comparison of Markov Chains and Random Fields on four different collections of polyphonic music. For Markov Chains we show the N-gram model that gave best performance on the testing set. For Random Fields we specify the number of iterations of the induction algorithm. For every collection, Random Fields result in lower testing perplexity and higher area under the ROC curve

The field of statistical music modeling is relatively new, and only a handful of models have been proposed for the task. Modeling polyphonic music is particularly challenging, and most existing models have focused on a much simple case of monophonic music. Accordingly, we feel that our work is an important contribution to the field.

7. REFERENCES

[1] D. A. and M. Clements. Melody spotting using hidden markov models. In J. S. Downie and D. Bainbridge, editors, *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pages 109–117, Indiana University, Bloomington, Indiana, October 2001.

[2] J. P. Bello, L. Daudet, and M. B. Sandler. Time-domain polyphonic transcription using self-generating databases. In *Proceedings of the 112th Convention of the Audio Engineering Society*, Munich, Germany, May 2002.

[3] 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.

[4] S. Chen, D. Beeferman, and R. Rosenfield. Evaluation metric for language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

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

[6] 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.

[7] L. Grubb and R. Dannenberg. Enhanced vocal performance tracking using multiple information sources. In *Proceedings of the ICMC*, pages 37–44, 1998.

[8] http://www.musedata.org. The musedata collection, 2000. Center for Computer Assisted Research in the Humanities (Stanford, CA).

[9] 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.

[10] K. MacMillan, M. Droettboom, and I. Fujinaga. Gamera: A structured document recognition application development environment. In J. S. Downie and D. Bainbridge, editors, *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pages 15–16, Indiana University, Bloomington, Indiana, October 2001.

[11] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *6th Workshop on Comp. Language Learning (CoNLL-2002)*, 2002.

[12] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML 2000*, pages 591–598, Stanford, California, 2000.

[13] G. Monti and M. Sandler. Pitch locking monophonic music analysis. In *Proceedings of the 112th Convention of the Audio Engineering Society (AES)*, Munich, Germany, May 10-13 2002.

[14] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.

[15] K. Papineni, T. Ward, and S. Roukos. Maximum likelihood and discriminative training of direct translation models. In *Proceedings of ICASSP-98, vol I, Seattle*, pages 189–192, 1998.

[16] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In M. Fingerhut, editor, *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)*, pages 140–149, IRCAM - Centre Pompidou, Paris, France, October 2002.

[17] C. Raphael. Automatic transcription of piano music. In M. Fingerhut, editor, *Proceedings of the 3rd Annual International Symposium on Music Information Retrieval (ISMIR)*, pages 15–19, IRCAM - Centre Pompidou, Paris, France, October 2002.

[18] R. Rosenfeld. A whole sentence maximum entropy language model. In *IEEE Workshop on Speech Recognition and Understanding Santa Barbara, CA*, 1997.

[19] A. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In *Proceedings of ACM International Multimedia Conference (ACMMM)*. ACM, ACM Press, 1998.

[20] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *Proceedings of ACM International Multimedia Conference (ACMMM)*, Orlando Florida, USA, Oct. 1999. ACM, ACM Press.

[21] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Trans. Information Theory*, 37:1085–1094, 1991.