Comparing models of symbolic music using probabilistic grammars and probabilistic programming

Samer A. Abdallah
University College London
s.abdallah@ucl.ac.uk

Nicolas E. Gold
University College London
n.gold@ucl.ac.uk

ABSTRACT

We conduct a systematic comparison of several probabilistic models of symbolic music, including zeroth and first order Markov models over pitches and intervals, a hidden Markov model over pitches, and a probabilistic context free grammar with two parameterisations, all implemented uniformly using a probabilistic programming language (PRISM). This allows us to take advantage of variational Bayesian methods for learning parameters and assessing the goodness of fit of the models in a principled way. When applied to a corpus of Bach chorales and the Essen folk song collection, we show that, depending on various parameters, the probabilistic grammars sometimes but not always out-perform the simple Markov models. On looking for evidence of overfitting of complex models to small datasets, we find that even the smallest dataset is sufficient to support the richest parameterisation of the probabilistic grammars. However, examining how the models perform on smaller subsets of pieces, we find that the simpler Markov models do indeed out-perform the best grammar-based model at the small end of the scale.

1. INTRODUCTION

Music usually exhibits a great deal of what is loosely called 'structure', both in acoustic and symbolic form, and over both local and global time scales. What exactly we mean by 'structure' could be discussed at great length, for example, in terms of relationships between parts, repetition, transformation, variation, not to mention specifically music-theoretic concepts such as scale, harmony, tonality and so on. One rather general approach to the notion of structure grew out Shannon's information theory [1], which prompted some psychologists [2, 3] to suggest that our perceptual systems are attuned to the detection of *redundancy* (an information theoretic concept) in sensory signals. Redundancy occurs when a sensory signal tells us something we already know, either from prior experience or from another part of the same sensory signal. Given the probabilistic underpinnings of information theory, this means, essentially, that redundancy is *any departure from complete unpredictability*, and this, we would argue, a good definition of what we mean by 'structure'. As animals in the world, detecting redundancy, especially over time, is a good thing to do, as it enables us to make predictions about what is coming next and to prepare for it accordingly. It has been said that 'the purpose of perception is to make the world seem less surprising'; in learning not to be surprised by a thing, we are forced to notice all the ways in which certain parts or aspects of that thing tell us about its other parts or aspects. This is the basis of our sense of its structure.

These ideas lead us firmly to probabilistic modelling as a way to understand both perception and structure, and therefore, by extension, *musical* structure; indeed, probabilistic modelling has been at the heart of music informatics for the last ten years or more, and before that, the role of expectation in shaping the listening experience had long been recognised [4, 5].

In the following sections, we will discuss some of the issues around probabilistic modelling (§ 2), probabilistic models of symbolic music (§ 3), and the use of probabilistic programming as a flexible environment in which to develop such models (§ 4). We will then describe the particular models we implemented and present the results of fitting these models in a number of variations to a corpus of symbolic music. We will take, for the sake of brevity, a rather less than rigorous approach to writing probability formulae, and will use the convention that, for example, $P(x|y)$ denotes for a conditional probability where x and y are the values of random variables left implicit given the context.

2. PROBABILISTIC MODELLING AND MODEL SELECTION CRITERIA

A probabilistic model of a domain is basically an assignment of probabilities to things that may occur in that domain. It is these probabilities which determine how surprising a thing is and, in temporal domains, how it might continue. In the sequel, when we say 'model', we mean *probabilistic* model. The discussion of models and data may seem somewhat removed from musical matters, but the reader should bear in mind that in application, 'data' becomes one or many pieces of music in symbolic form, and a 'model' can embody anything from a structural analysis of a single piece to an entire music theory for a large corpus. Hence, this section is concerned with answering the questions, 'how do we recognise a good music analysis when we see one?'; 'how do we recognise a good music theory when we see one?'

In order to be able to adapt to new situations (for example, new styles of music), we must be able to build or adapt models on the basis of observations. Even when listening to an individual piece of music, our expectations are fluid and adaptable: a theme or motif is less surprising in reprise than on initial presentation. When there is only a finite amount of data, it is not possible to pin-down a single 'correct' model, and we need a way to evaluate candidate models against each other to establish which one is likely to give the best predictions. In machine learning, this is the problem of model selection, and brings up a number of issues which are familiar in that field. The use of an overly complex model with many parameters that must be inferred from the data can result in *over-fitting* when applied to too small a dataset, resulting in poor *generalisation*, that is to say, the model becomes tightly coupled to the intimate, accidental details of the data and fails to recognise (is surprised by) more data of the same sort. On the other hand, an overly simple model might not be capable of capturing the regularities that are in the data. Some way of managing the trade-off is required.

Although there are other methods (such as cross-validation), Bayesian model selection criteria offer a theoretically and philosophically appealing solution to this problem [6, 7]. The fundamental basis of Bayesian inference is the consistent use of probabilities to represent the uncertainty of the agent doing the inferring about all entities under consideration, including models and their parameters. [1] For example, given an agent committed to a parameterised model $\mathcal{M}$, the agent does not initially know how the parameters θ should be set, and must represent this uncertainty as a *prior* probability distribution $P(\theta|\mathcal{M})$. Then, on observing some data $\mathcal{D}$, the agent should update its 'belief state', giving, not a point estimate of θ, but a *posterior* probability distribution

$$P(\theta|\mathcal{D},\mathcal{M}) = \frac{P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}, \qquad (1)$$

which takes into account both the prior and the likelihood that the model with parameters θ could have produced the observed data. The reason why this is the correct policy is that, in order to make the best possible prediction of a new piece of data d, given the model and observations so far, the agent needs to compute

$$\begin{aligned} P(d|\mathcal{D},\mathcal{M}) &= \int P(d,\theta|\mathcal{D},\mathcal{M})\ \mathrm{d}\theta \\ &= \int P(d|\theta,\mathcal{M})P(\theta|\mathcal{D},\mathcal{M})\ \mathrm{d}\theta. \end{aligned}$$

This means the agent can forget about the data $\mathcal{D}$ as long as it remembers the posterior distribution $P(\theta|\mathcal{D},\mathcal{M})$. The denominator in (1) is known as the *evidence* and can be computed as

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})\ \mathrm{d}\theta. \qquad (2)$$

If there are several candidate models $\mathcal{M}_1,\ldots,\mathcal{M}_N$, then the whole inferential process is lifted from distributions over parameters to distributions over models, with prior $P(\mathcal{M}_i)$ and posterior

$$P(\mathcal{M}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathcal{D})}. \qquad (3)$$

If the agent is initially uncommitted to any particular model, so $P(\mathcal{M}_i)$ is relatively flat, then we can see that the evidence $P(\mathcal{D}|\mathcal{M})$ plays the key role in determining the relative plausibility of the models after the data has been observed. The committed Bayesian will work with this posterior distribution to make predictions and decisions (this is *model averaging*), but forced to make a choice, perhaps because of limited computational resources for keeping track of multiple models, a reasonable policy is to pick the model with the greatest evidence.

If a model is too simple, then it may not be able to fit the data and ends up assigning low probability to our given dataset $\mathcal{D}$, resulting in low evidence. If it is too complex, then it may be able to fit our given dataset well, but also many other possible datasets. It assigns significant probability to a greater variety of datasets and therefore less to any particular dataset, also reducing the evidence. Hence, using the evidence as the model selection criterion *automatically* penalises models which are more complex than the data can support, giving a formal expression of Ockam's razor, the philosophical principle that, other things being equal, we should choose the *simplest* explanation.

Another view on simplicity is provided by the *minimum message length* principle [9] (and the related *minimum description length* [10]) which states that we should adopt a model that allows us to produce the shortest possible description of the data, including the description of any model parameters. However, given the close relationship between compression and probabilistic structure, this leads to a conclusion which is essentially the same as the Bayesian approach [11], modulo some minor differences [12].

Representing uncertainty about model parameters, computing the evidence and doing model averaging can be expensive operations computationally and approximations are often needed. For some models, *variational Bayesian learning* [13, 14] can be a good solution, combining an efficient representation of uncertainty about parameters with a tractable learning algorithm, delivering an estimate of the evidence, as a function of the *variational free energy* F. We omit the details of how this is defined, but note that the algorithm works to minimise F by adjusting its approximation of the posterior (1), and F is an upper bound on $-\log P(\mathcal{D}|\mathcal{M})$, that is,

$$-\log P(\mathcal{D}|\mathcal{M}) \le F, \qquad (4)$$

and so, after learning is complete, we can use F instead of the true evidence for model comparisons. Thus, we come to the methodology we adopt for our subsequent modelling experiments: given a dataset and number of candidate models, we fit each model using variational Bayesian learning and use the variational free energy to compare them: the lower the free energy, the better the model.

[1] If we choose to represent degrees of belief as real numbers, probability theory is the *only* consistent method for reasoning under uncertainty [8].

3. MODELLING SYMBOLIC MUSIC

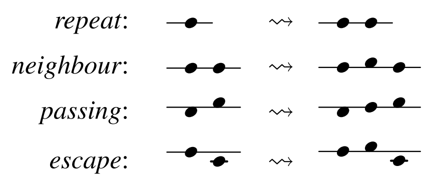
Probabilistic models of symbolic music can, to a large extent, be divided into two broad classes: those based on Markov (n-gram) models, and those based on grammars. While fixed-order n-gram models have problems avoiding over-simplicity for low n and over-fitting for high n, *variable* order Markov models have been used successfully to model monophonic melodic structure [15, 16] and chord sequences [17]. We review grammar-based models below and discuss the relationship between the two classes in § 3.2.

3.1 Grammar-based models

Formally, a context free grammar (CFG) consists of a set of *terminal* symbols, a set of *non-terminal* symbols, a set of production rules describing how each non-terminal can be rewritten as a sequence of terminals or non-terminals, and a distinguished non-terminal called the start symbol. A probabilistic CFG (PCFG) adds to this a probability distribution over the possible expansions of each non-terminal.

Grammars have been applied in computational musicology since the late 1960s [18, 19, 20], resulting in influential theories like Lerdahl and Jackendoff's *Generative Theory of Tonal Music* [21] and Steedman's jazz chord sequence grammar [22]. However, *probabilistic* grammar-based models of music are a relatively recent development, broadly falling into two groups: models of *harmonic* sequence [23, 24], and models of *melodic* sequence [25, 26, 27]. We focus on melodic models only in this paper.

Gilbert and Conklin [26] applied a PCFG to melodic structure analysis, drawing parallels between their approach and the hierarchical graphs of Schenkerian analysis [28], that also attempts to account for the details of melodic structure in terms of elaborations of simpler underlying forms. Schenker's elaborations are similar to grammar production rules, but because some of them (such as the introduction of neighbour notes or passing notes) depend on *two* adjacent notes, they cannot be written as a *context free* grammar if the melody is represented as a sequence of pitches. By representing melody as a sequence of pitch *intervals*, Gilbert and Conklin were able to devise a CFG that embodies four type of melodic elaboration illustrated thus (as in figure 1 of [26]):



Mavromatis and Brown [29] reported that they were able to design a CFG for Schenkerian analysis by adopting the same policy of elaborating interval sequences. Kirlin and Jensen [27] also base their probabilistic model of musical hierarchies on the elaboration of intervals, but adopt Yust's [30] triangulated graphs as their structured representation, rather than the trees of conventional grammatical analysis.

Other attempts to formalise Schenkerian analysis that are not explicity probabilistic but do incorporate heuristics to guide the search for reductions include work by Ebcioğlu [31] and Marsden [32]. Marsden in particular discusses how adopting an interval-based encoding, to avoid the need for context-dependent rules, leads to problems if the grammar is developed further to model durations and to cover more types of elaboration. [2] Nonetheless, we will base our probabilistic grammar on that of Gilbert and Conklin.

3.2 Markov- *vs* Grammar-based models

The contrast between Markov-based and grammar-based approaches reflects a similar division in computational linguistics, where probabilistic grammars and statistical parsing are widely used for tasks where the hierarchical structure of a syntactic analysis is required (e.g. language understanding). Such approaches do not perform as well as n-gram models in assigning probabilities to sentences. This was discussed in 1998 by Brill *et al* [33], and, while n-gram and grammar based models have both advanced since then, variable order Markov models continue to out-perform grammatical models (e.g. [34]). It appears that what Markov models lack in linguistic sophistication they more than make up for in the raw statistical power of learning which words tend to occur together regardless of syntax. In computational musicology, a systematic framework for comparing across models has yet to be established despite increasing research activity using probabilistic grammars.

We propose that such a framework can be provided using variational Bayesian learning within a probabilistic programming language able to support a wide variety of models. We have begun with small number of relatively simple models, but the framework will support the exploration of increasingly sophisticated models such that robust musicological conclusions can be drawn about their relative merits when applied to a variety of musical corpora.

4. IMPLEMENTATION

Probabilistic programming Probabilistic programming languages aim to provide an environment where a wide variety of probabilistic models can be defined succinctly and in a way that goes beyond such formalisms as Bayesian networks, by making available powerful constructs that are familiar from ordinary programming languages, such as abstraction, recursion, and structured data types. The earliest grew out of probabilistic logics developed in the logic programming community [35, 36, 37], but very soon an alternative branch of the family was developed based on concepts from functional programming [38, 39]. More recently, interest in this area has grown quite considerably and there are many languages, each exploring various aspects and implementation strategies, for example [40, 41, 42].

We adopted a language called PRISM (PRogramming In Statistical Modelling), which has been in development since 1995 [36]. As it is based on Prolog, it inherits logic programming features such as logic variables and powerful meta-programming facilities.

[2] Even with the *repeat* elaboration, there are subtleties to consider: should an interval of N semitones be expanded to $0\,N$, or $N\,0$? The answer relates to whether we associate the interval with the time-span of the first or the second note in the interval respectively. Gilbert and Conklin follow the *latter* convention; we, in our grammar, the *former*.

PRISM was chosen because it has several features that make it suitable for implementing probabilistic grammars. Firstly, because it inherits Prolog's definite clause grammar (DCG) notation, CFGs can be encoded very succinctly. Secondly, because it inherits and relies on the underlying Prolog's tabling mechanism, the process of parsing replicates the structure and computational complexity of Earley's efficient chart parsing algorithm without any special effort by the programmer [43]. [3] Thirdly, PRISM provides parameter learning mechanisms that subsume standard expectation maximisation (EM) and variational Bayesian (VB) methods [44, 40]. This enables us to compare models on the basis of variational free energy, as discussed in § 2.

PRISM has been used for implementing probabilistic gramars for natural languages and estimating their parameters [45] and for doing grammar induction using VB for model selection [44]. PRISM has also been used for music modelling, but using a hidden Markov model rather than a grammar-based model [46].

4.1 Implementing PCFGs in PRISM

A PCFG can be easily implemented in PRISM by writing a DCG interpreter with probabilistic choice between alternative rule head expansions. In ordinary Prolog, DCG rules (non-terminals) can be parameterised and arguments used to represent such linguistic phenomena as number or tense agreements [4] . Augmenting a DCG with probabilities presents some difficulties which become apparent in the implementation. The problem is that *unification*, inherent in DCG processing, amounts to the imposition of constraints and can result in failure. Introducing failure into a probabilistic program results in a significant complication of inference and learning [47].

For our purposes, the problem of failure can be avoided by breaking the rule expansion process into two stages: first, a given non-terminal is matched against all applicable rule heads, and for each rule that matches, its optional *guard* (an ordinary Prolog goal) is executed. Rules with successful or absent guards are collected and then chosen from probabilistically. The bodies of these rules are not allowed to fail. Any constraints which might cause failure must be encoded in the rule heads or the guards. Thus our DCG language is similar to standard Prolog DCGs but, instead of the usual *Head* $\longrightarrow$ *Body* notation, rules are written in one of two forms (notes on Prolog syntax can be found in appendix A):

$$\textit{Head} :: \textit{Label} \implies \textit{Body}.$$
$$\textit{Head} :: \textit{Label} \implies \textit{Guard}\,|\textit{Body}.$$

Label is an atom that is unique for different clauses of the same nonterminal, and *Guard* is an ordinary Prolog goal that can only use variables in *Head*. A terminal symbol *X* is written out using *+X* instead of [*X*], and *nil* is used instead of [] for an empty rule body. Finally, PRISM switches, which are PRISM's primitive for probabilistic choice and represent learnable probability distributions, can be sampled using goals of the form *Val~Switch*, which corresponds to the PRISM goal *msw*(*Switch*,*Val*). The range of values for a given switch is determined by a corresponding *values*(*Switch*,*Values*) clause.

All of this can be illustrated with reference to the program in fig. 3. The neighbour note rule (labelled *neigh*) applies to the expansion of a non-terminal *i*(*P*), where *P* is a pitch interval in semitones, but only when *P*=0. The deviation *P1* to the neighbour note is sampled from a random switch called *step*, and is between -4 and 4. The rule labelled *term* shows how a non-terminal *i*(*P*) can produce a terminal symbol, in this case, the integer *P*.

Parameterisation of rule expansion distributions Programs written in our DCG language define the permissible expansions for non-terminals with arguments, but not how the probability distributions over those expansions are parameterised. We implemented two approaches. The first is to treat each ground instance of each rule (that is, with definite values for all variables) as an independent PCFG rule with its own distribution that can be learned from examples. This corresponds to the "rule schema" approach adopted by Gilbert and Conklin, and we will refer to it as the "ground head" parameterisation.

An alternative is to collect together all rule heads with the same functor and arity *and* which lead to the same set of applicable expansions, and have them share a single probability distribution. For example, in fig. 3, all non-terminals of the form *i*(*P*) where the absolute value of *P* is between 6 and 16 share the functor *i*/1 and can be expanded using the rules *term*, *rep* and *esc*. Thus, under this "head functor" parameterisation, they share the same probability distribution over those three expansion rules. This approach generally produces a model with fewer parameters, which could potentially reduce the likelihood of over-fitting to small datasets.

5. EXPERIMENTS

Using the implementation framework described above, we conducted an experiment to compare the performance of several models on a corpus of monophonic melodies. We used a corpus of scores in Humdrum/Kern format, comprising three datasets, all available from the Kern Scores website at *http://kern.humdrum.org*. The first is a set of 185 Bach chorales, BWV 253–438 excluding 279. This is the dataset that was used by Gilbert and Conklin. The second is a larger set of 370 Bach chorales. The third is the Essen folk song collection, containing 6174 scores. Because the full Essen collection was too large to process with the Gilbert and Conklin grammar on our test computer (an Apple laptop with 8 GB of memory), we took two random subsets of 1000 scores each. These datasets will be referred to as *chorales*, *chorales371*, *essen1000a* and *essen1000b* respectively.

5.1 Methods

A total of seven probabilistic models were implemented as PDCGs. Pitches are encoded as MIDI note numbers, while interval-based models assume an input encoded as a list of integers followed by the Prolog atom *end*. This is

[3] For a variety of models, tabling results in efficient probability computations equivalent to the optimal special purpose algorithms for those models, such as the forwards-backwards algorithm for HMMs, the inside-outside algorithm of PCFGs, and belief propagation in Bayesian networks.

[4] Indeed, the DCG formalism is Turing complete and can therefore, in principle, represent any linguistic phenomenon.

```
values(nnum, X) :− numlist(40,100,X).
values(mc(_), X) :− values(nnum,X).
values(hmc(_), X) :− num_states(N), numlist(1,N,X).
values(obs(_), X) :− values(nnum,X).

% start symbol for p1gram
s0 :: tail  ⟹ nil.
s0 :: cons  ⟹ X~nnum, +X, s0.

% start symbol for p2gram
s1(_) :: tail  ⟹ nil.
s1(Y) :: cons  ⟹ X~mc(Y), +X, s1(X).

% start symbol for phmm
sh(_) :: tail  ⟹ nil.
sh(Y) :: cons  ⟹ X~hmc(Y), Z~obs(X), +Z, sh(X).
```

Figure 1. PDCGs for 0th and 1st order Markov chains and 1st order HMMs over pitch (encoded as MIDI note number). The number of states in the HMM is a parameter of the model.

```
values(ival, X) :− numlist(−20,20,X).
values(mc(_), X) :− get_values(ival,X).

% start symbol for i1gram
s0 :: tail  ⟹ +end.
s0 :: cons  ⟹ X~ival, +X, s0.

% start symbol for i2gram
s1(_) :: tail  ⟹ +end.
s1(Y) :: cons  ⟹ X~mc(Y), +X, s1(X).
```

Figure 2. PDCG for 0th and 1st order Markov chains over pitch interval to next note in semitones.

because we chose to represent each note by the pitch interval to the *following* note, and the last note has no following note. As we develop the models to handle other musical dimensions (e.g. duration, metrical strength, articulation etc.), the attributes of the last note can be associated with the *end* symbol. The models, with their short names, are:

1. 0th order Markov model over pitches (*p1gram*).
2. 1st order Markov model over pitches (*p2gram*).
3. 1st order hidden Markov model over pitches (*phmm*).
4. 0th order Markov model over intervals (*i1gram*).
5. 1st order Markov model over intervals (*i2gram*).
6. Modified Gilbert and Conklin grammar with grounded head parameterisation (*gilbert2*).
7. Modified Gilbert and Conklin grammar with head functor parameterisation (*gilbert3*).

The DCG rules for all of these models are shown in fig. 1 (*p1gram*, *p2gram* and *phmm*), fig. 2 (*i1gram* and *i2gram*), and fig. 3 (*gilbert2* and *gilbert3* share the same rules and differ only in their parameterisation). We have omitted some supplementary code for intialising the switch probabilities and other ancillary tasks, as well as the DCG interpreter itself; these are available from the authors on request.

```
values(step, X) :− numlist(−4,4,X).
values(leap, X) :− numlist(−16,16,X).
values(passing(N), Vals) :−
    ( N>0 → M is N−1, numlist(1,M,I1)
    ; N<0 → M is N+1, numlist(M,−1,I1)
    ),
    maplist(N1, (N1,N2),N2 is N−N1,I1,Vals).

values(escape(N), Vals) :−
    ( N<0 → I1 = [1,2,3,4]
    ; N>0 → I1 = [−1,−2,−3,−4]
    ),
    maplist(N1,(N1,N2),N2 is N−N1,I1,Vals).

% start symbol
s :: last  ⟹ i(end).
s :: grow  ⟹ P~leap, i(P), s.

i(P) :: term   ⟹ +P.
i(P) :: rep    ⟹ i(0), i(P).
i(P) :: neigh  ⟹ P=0 |
                  P1~step, {P2 is −P1}, i(P1), i(P2).
i(P) :: pass   ⟹ passable(P) |
                  (P1,P2)~passing(P), i(P1), i(P2).
i(P) :: esc    ⟹ escapable(P) |
                  (P1,P2)~escape(P), i(P1), i(P2).

passable(P) :− abs_between(2,5,P).
escapable(P) :− abs_between(1,16,P).
abs_between(L,U,X) :− Y is abs(X), between(L,U,Y).
```

Figure 3. A grammar modelled on Gilbert and Conklin's [26], written in a DCG language defined in PRISM. *maplist/5* and *between/3* are standard B-Prolog predicates and *numlist(L,U,X)* is true when *X* is a list of consecutive integers from *L* to *U*.

All the Markov models use the head functor parameterisation, so for *s0*, *s1*(_) and *sh*(_) there is only one distribution over the labels [*tail*, *cons*] which determines whether or not the chain is terminated or continues. The transition distributions are determined by the PRISM switches *mc*(_) and *hmc*(_), and *are* distinct for each ground instance, as are the observation distributions for the HMM.

Our version of Gilbert and Conklin's grammar differs from the original in two ways. Firstly, it uses a different mechanism for introducing new intervals not explained by any of the elaboration rules: the *s* non-terminal is not parameterised by interval and simply expands into a sequence of *i*(*P*) non-terminals, where the *P* are intervals chosen independently from the *leap* distribution. Secondly, because an interval is associated with the *former* note of a pair, the *rep* elaboration maps *i*(*P*) to *i*(0), *i*(*P*) and not *i*(*P*), *i*(0) as in Gilbert and Conklin's version. In our grammar, this has the sense of subdividing the note with the repeated pitch, whereas in Gilbert and Conklin's, it is the *preceding* note that is subdivided. This is clearer if one imagines that a note has a duration which is also subdivided along with the pitch interval. It would be possible to compare the two approaches directly within this framework, but we have not done this yet. Finally, the numerical ranges of steps, leaps, and the limits for allowing passing or escape note introductions had to be chosen arbitrarily as this information was not given in Gilbert and Conklin's paper.

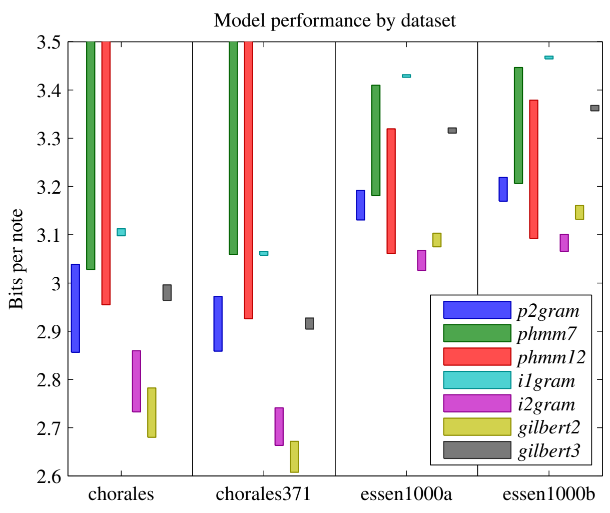


Figure 4. Overall performance, in bits per note, of each model against each dataset. The smaller the bpn is, the better the performance. For each model/dataset pair, the bar shows the range values obtained over the various parameter combinations described in the text. The *phmmN* bars represent the HMMs with N states; the bars for the chorales datasets extend off the top of the chart.

Parameterisation A number of 'hyperparameters' control the Dirichlet priors over the probability distributions for each switch, affecting the *shape* of distributions (e.g. over pitch intervals or absolute pitches) that might be expected to be weighted towards a central value (e.g. zero in the case of intervals, or a central register for absolute pitches). The distributions are given their expected shape by a weighted sum of binomial and uniform distributions. The *prior_weight* hyperparameter affects all distribution priors, effectively determining the volume of data required to override prior beliefs about the switch distribution. The complete set of hyperparameters and their range of values is shown below.

prior_weight:	$\{0.3, 1, 3, 10, 30\}$.	**% all models**
prior_shape :	*shape_spec*	**% for all Markov models**
leap_shape :	*shape_spec*	**% for grammar models**
pass_shape :	*shape_spec*	**% for grammar models**
num_states :	$\{1, 2, 3, 5, 7, 12\}$	**% for HMMs**
trans_self :	$\{0, 1\}$	**% for HMMs**

$$\mathit{shape\_spec} = \{\mathit{binomial},\ \mathit{uniform},\ \mathit{binomial{+}uniform}\} \cup \{\mathit{binomial} + K{*}\mathit{uniform}\ |K\ \mathit{in}\ \{0.1, 0.3\}\}$$

Subset selection We extracted subsets of elements from each dataset to evaluate the effect of dataset size on comparative model performance and to investigate whether the complex models over-fit to small datasets. Subsets of size 1 to 30 were considered, 10 subsets being chosen at random from the full dataset in each case. The same subsets were supplied to all models for training. Thus, each subset can be identified by a dataset name, a size from 1 to 30, and an index from 1 to 10.

5.2 Results

Fig. 4 shows a summary of the overall performance of each model on each dataset, under a range of parameter values. To compare performance between datasets of different sizes, the variational free energy in each case is divided by the total number of notes in the dataset and displayed in 'bits per note' (bpn). The data for *p1gram*, the 0th order Markov model over pitches, is not shown, as its best-case performance was always worse than that of all the other models, achieving at best about 3.7 bpn on the chorales.

The next-worst model is *i1gram*. It performs worse than *p2gram*, consistent with the fact that a pair of consecutive pitches (a 2-gram) contains information about both pitch interval *and* absolute pitch, while the latter is not available to *i1gram*. If sparsity and over-fitting can be avoided, *p2gram* should be able to make predictions at least as well as *i1gram*. The results show that even the smallest dataset *chorales* is large enough to permit this.

The HMMs have the widest range of results, most likely because the learning algorithm has a tendency to get stuck in local optima. The HMMs, for larger state-space sizes, perform noticably better on Essen collection than on chorales.

Proceeding onwards, the *p2gram* and *gilbert3* models overlap somewhat for the chorales, but not for the Essen collection. Under their best (respective) parameter settings, *p2gram* performs better than *gilbert3* for all datasets. Perhaps surprisingly, the PCFG in its less flexible (head functor parameterisation) form is out-performed by a first order Markov model over pitches, showing that one cannot assume that a grammar-based model will always out-perform even a first-order Markov model.

For all datasets, the best two consistently performing models are *gilbert2* and *i2gram*. The latter achieves approximately 2.68 bpn on the *chorales* dataset with the parameter settings:

prior_weight :	3,
leap_shape :	*binomial* + 0.1∗*uniform*,
pass_shape :	*binomial*,

though the results are fairly insensitive to the *pass_shape* parameter. This is comparable with the 2.67 bpn reported by Gilbert and Conklin (bear in mind that the variational free energy includes a model complexity penalty). It is encouraging to note that the grammar-based model gives the best account of both chorales datasets, although it is beaten by the Markov model on the larger Essen datasets. Considering that higher-order Markov models would be likely to perform better still (since the Essen collection is large enough to support a more complex model) this shows that designing by hand a probabilistic grammar capable of out-performing variable order Markov models is a non-trivial task.

Fig. 5 shows how the models perform relative to each other on the much smaller datasets obtained by extracting random subsets from the *chorales* dataset. The graph shows how, for smaller datasets, the simpler Markov models out-perform the grammar-based models, with *gilbert2* only emerging as best with datasets of 20 or more pieces. The first order Markov model over intervals, *i2gram*, though out-performed for larger datasets, performed consistently well over the whole range.

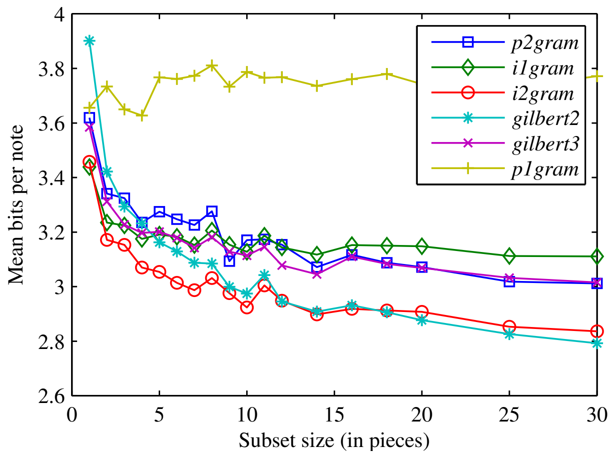


Figure 5. Variation of model performance with dataset size. For each subset size K, 10 random subsets of size K were extracted from the *chorales* dataset. Then, for each model, the parameter settings obtaining the best mean performance over the 10 subsets were determined, and this mean performance was plotted against the subset size. Though there is some random variation due to the random subset selection mechanism, the graph clearly shows how the most complex model, *gilbert2*, performs the worst for the smallest datasets, even compared with *p1gram*.

6. DISCUSSION AND CONCLUSIONS

We have shown how a variety of probabilistic models of symbolic music can be implemented in a framework of probabilistic programming, and applied these to collections of Bach chorales and the Essen folk song collection. It was found that a probabilistic grammar-based on that of Gilbert and Conklin [26] performed best on the Bach chorales, acheiving performance comparable to that acheived by them on the *chorales* dataset. However, an alternative, more parsimonious parameterisation of the same grammar performed worse than a 1st order Markov model over pitch intervals. On the Essen folksong collection, the grammar-based model was beaten by the 1st order Markov model over intervals. This may be due to the greater stylistic uniformity of the chorales datasets and will be investigated in future work. The relative success of the HMMs on the Essen collection is also notable and worthy of investigation.

Focussing on the chorales, we found that, as the size of the dataset decreases, the 1st order Markov model over intervals begins to out-perform the grammar-based model, until, for very small datasets, the 0th order Markov model over intervals performs best. This highlights the need to consider modelling assumptions carefully when dealing with small collections of music, which may often be the case when conducting an analysis of certain stylistically related pieces, of which only a small number may exist.

It is likely that variable order Markov models will improve significantly on the performance of *i2gram*, challenging us to develop better grammar-based models. Our framework and initial models provide a basis for such developments, and a systematic exploration of probabilistic models of music in both analysis and the development of music theory.

Acknowledgments

This work is supported by the UK Engineering and Physical Sciences Research Council CREST Platform project [grant number EP/G060525/2] and UK Arts and Humanities Research Council Digital Music Lab project [grant number AH/L01016X/1]. Data may be obtained from either of the authors subject to copyright restrictions.

7. REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423,623–656, 1948.

[2] F. Attneave, "Some informational aspects of visual perception," *Pyschological Review*, vol. 61, no. 3, pp. 183–193, 1954.

[3] H. B. Barlow, "Sensory mechanisms, the reduction of redundancy, and intelligence," in *Proceedings of a Symposium on the Mechanisation of Thought Processes*, vol. 2. National Physical Laboratory, Teddington: Her Majesty's Stationery Office, London, 1959, pp. 537–559.

[4] E. Hanslick, *On the musically beautiful: A contribution towards the revision of the aesthetics of music*. Indianapolis, IN: Hackett, 1854/1986.

[5] L. B. Meyer, *Emotion and Meaning in Music*. University of Chicago Press, 1956.

[6] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 773–795, 1995.

[7] D. L. Dowe, S. Gardner, and G. Oppy, "Bayes not bust! Why simplicity is no problem for Bayesians," *The British Journal for the Philosophy of Science*, vol. 58, no. 4, pp. 709–754, 2007.

[8] R. T. Cox, "Probability, frequency and reasonable expectation," *American Journal of Physics*, vol. 14, pp. 1–13, 1946.

[9] C. S. Wallace and D. M. Boulton, "An information measure for classification," *The Computer Journal*, vol. 11, no. 2, pp. 185–194, 1968.

[10] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[11] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[12] R. A. Baxter and J. J. Oliver, "MDL and MML: Similarities and differences," Department of Computer Science, Monash University, Tech. Rep. 207, 1994.

[13] D. J. C. MacKay, "Ensemble learning for hidden Markov models," Cavendish Laboratory, Cambridge University, Tech. Rep., 1997.

[14] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, M. I. Jordan, Ed. Cambridge, MA: MIT Press, 1998, pp. 105–161.

[15] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[16] M. T. Pearce, "The construction and evaluation of statistical models of melodic structure in music perception and composition," Ph.D. dissertation, Department of Computing, City University, London, 2005.

[17] K. Yoshii and M. Goto, "A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis." in *ISMIR*, 2011, pp. 645–650.

[18] T. Winograd, "Linguistics and the computer analysis of tonal harmony," *Journal of Music Theory*, vol. 12, no. 1, pp. 2–49, 1968.

[19] M. Kassler, "A trinity of essays," Ph.D. dissertation, Princeton University, 1967.

[20] B. Lindblom and J. Sundberg, *Towards a generative theory of melody*. Department of Phonetics, Institute of Linguistics, University of Stockholm, 1970.

[21] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press, 1983.

[22] M. J. Steedman, "A generative grammar for jazz chord sequences," *Music Perception*, vol. 2, no. 1, pp. 52–77, 1984.

[23] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.

[24] M. Granroth-Wilding, "Harmonic analysis of music using combinatory categorial grammar," Ph.D. dissertation, School of Informatics, University of Edinburgh, 2013.

[25] R. Bod, "Probabilistic grammars for music," in *Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*, 2001.

[26] É. Gilbert and D. Conklin, "A probabilistic context-free grammar for melodic reduction," in *International Workshop on Artificial Intelligence and Music, IJCAI-07, Hyderabad, India*, 2007.

[27] P. B. Kirlin and D. D. Jensen, "Probabilistic modeling of hierarchical music analysis," *Analysis*, vol. 1, p. 15, 2011.

[28] H. Schenker, *Der freie Satz*. Vienna: Universal Edition, 1935, (Published in English as E. Oster (trans., ed.) *Free Composition*, Longman, New York, 1979.).

[29] P. Mavromatis and M. Brown, "Parsing context-free grammars for music: A computational model of Schenkerian analysis," in *Proc. 8th Intl. Conf. on Music Perception and Cognition, Evanston, USA*, 2004, pp. 414–415.

[30] J. Yust, "The geometry of melodic, harmonic, and metrical hierarchy," in *Mathematics and Computation in Music*. Springer, 2009, pp. 180–192.

[31] K. Ebcioğlu, "An expert system for harmonizing four-part chorales," *Computer Music Journal*, vol. 12, no. 3, pp. 43–51, 1988.

[32] A. Marsden, "Schenkerian analysis by computer: A proof of concept," *Journal of New Music Research*, vol. 39, no. 3, pp. 269–289, 2010.

[33] E. Brill, R. Florian, J. C. Henderson, and L. Mangu, "Beyond n-grams: Can linguistic sophistication improve language modeling?" in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 1998, pp. 186–190.

[34] F. Wood, C. Archambeau, J. Gasthaus, L. James, and Y. W. Teh, "A stochastic memoizer for sequence data," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1129–1136.

[35] D. Poole, "Representing Bayesian networks within probabilistic Horn abduction," in *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1991, pp. 271–278.

[36] T. Sato, "A statistical learning method for logic programs with distribution semantics," in *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, Tokyo, 1995, pp. 715–729.

[37] S. Muggleton, "Stochastic logic programs," *Advances in inductive logic programming*, vol. 32, pp. 254–264, 1996.

[38] D. Koller, D. McAllester, and A. Pfeffer, "Effective Bayesian inference for stochastic programs," in *AAAI/IAAI*, 1997, pp. 740–747.

[39] A. Pfeffer, "IBAL: A probabilistic rational programming language," in *IJCAI*. Citeseer, 2001, pp. 733–740.

[40] T. Sato, Y. Kameya, and K. Kurihara, "Variational Bayes via propositionalized probability computation in prism," *Annals of Mathematics and Artificial Intelligence*, vol. 54, no. 1-3, pp. 135–158, 2008.

[41] O. Kiselyov and C.-C. Shan, "Embedded probabilistic programming," in *Domain-specific languages*. Springer, 2009, pp. 360–384.

[42] N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. Tenenbaum, "Church: a language for generative models," in *Uncertainty in Artificial Intelligence*, 2008.

[43] H. H. Porter, "Earley deduction," Oregon Graduate Center, Tech. Rep., 1986.

[44] K. Kurihara and T. Sato, "Variational Bayesian grammar induction for natural language," in *Grammatical Inference: Algorithms and Applications*. Springer, 2006, pp. 84–96.

[45] T. Sato, S. Abe, Y. Kameya, and K. Shirai, "A separate-and-learn approach to EM learning of PCFGs." in *NLPRS*, 2001, pp. 255–262.

[46] J. Sneyers, J. Vennekens, and D. De Schreye, "Probabilistic-logical modeling of music," in *Practical Aspects of Declarative Languages*. Springer, 2006, pp. 60–72.

[47] T. Sato, Y. Kameya, and N.-F. Zhou, "Generative modeling with failure in PRISM." in *IJCAI*, 2005, pp. 847–852.

A. PROLOG SYNTAX

Prolog code and data consist of *terms* built from a *functor* and a number of arguments; e.g., $a(10,b,X)$ is a term with a head functor $a/3$ (because it has three arguments), and arguments 10 (an integer), b (an atom or symbol), and X (a logic variable). Atoms and functor names start with a lower-case letter, while variable names start with an upper-case letter or underscore. A solitary underscore (_) stands for a variable whose value is not needed. Functors can be declared as prefix, infix, or suffix operators, for example, we declare ~ to be an infix operator, so the head functor of *P~leap* is ~/2. The definite clause grammar (DCG) notation allows grammar rules to be defined using clauses of the form *Head* $\longrightarrow$ *Body*, where *Head* is a term and *Body* is a list of one or more comma separated DCG goals. Within the body, a list [X,Y,...] represents a sequence of terminals, while a term enclosed in braces, {*Goal*} is interpreted as an ordinary Prolog goal.