Exploring probabilistic grammars of symbolic music using PRISM

Samer Abdallah and Nicolas Gold

Department of Computer Science, UCL

PLP Workshop, Vienna
July 17, 2014

Outline

Outline

The main idea

To use probabilistic grammars for analysing music.

Repeat as necessary:

1. Design or otherwise obtain (adapt, grow, evolve etc.) a probabilistic grammar of music.
2. Reality check: is model sufficiently unsurprised by your test corpus? (i.e., does the model fit the data better than previous efforts?)
3. Parse music with grammar to obtain a probability distribution over parse trees, or maybe just the top few most probable parses.
4. Interpret parse trees as an analysis.

Why do this?

Before this kind of technology was invented, the only way to get an analysis of a piece of music was to find a musicologist.

Music → **Musicologist** → **Analysis**

Failing this, a music student might do, or someone who listens to a lot of that sort of music; possibly you could do it yourself.

There are problems with this and it and they raise a lot of questions…

Why do this?

Problems and questions:

- It might take a long time to get an analysis.
- There aren't that many musicologists around.
- Even if you can find a music student or do it yourself, how do you know he/she/you have done a good job? What does that even mean?
- Even amongst "experts" there can be a lot of variability in the analyses they produce.
- Musicologists are very complex. There's a lot of stuff going on in there that we don't understand very well.

(And of course, all of this begs the question, why analyse music at all?)

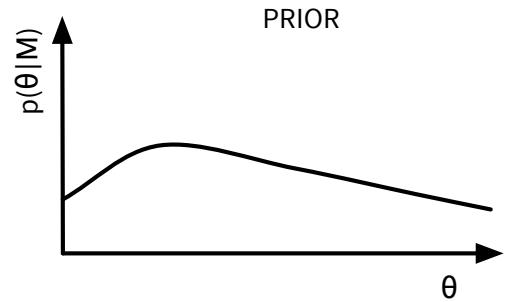
One way forward is to try to find some general principles that govern how humans react to complex objects like music.

Outline

Learning parameteric models

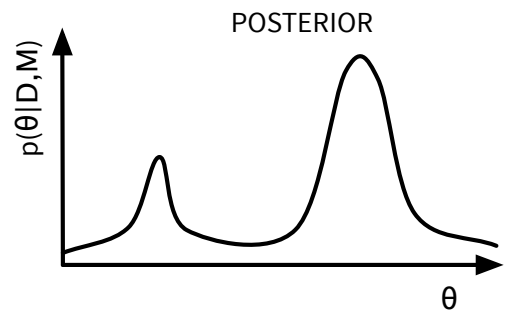
Suppose we have some data $\mathcal{D} = (d_1, \ldots, d_T)$ and wish to understand it with model $\mathcal{M}$ which has some parameters θ. Model assigns probabilities $P(d_i|\theta, \mathcal{M})$ to items d_i and assumes items are independent given model and parameters, so the *likelihood* is:

$$P(\mathcal{D}|\theta, \mathcal{M}) = \prod_{i=1}^{T} P(d_i|\theta, \mathcal{M}).$$



The, prior is $P(\theta|\mathcal{M})$ and posterior is

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



Why is this the right thing to do? Because *the posterior distribution contains all the information in the data that is required to make predictions*.

Bayesian evidence

The denominator in (1) is known as the *evidence* and can be expressed as

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

It measures how suprising the data was as far as that model is concerned, and becomes useful later for comparing models.

Bayesian Model selection

Now suppose we have several candidate models $\mathcal{M}_1, \ldots, \mathcal{M}_N$ to consider. Then do Bayesian inference over model identity: start with a prior $P(\mathcal{M}_i)$ and compute the posterior

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

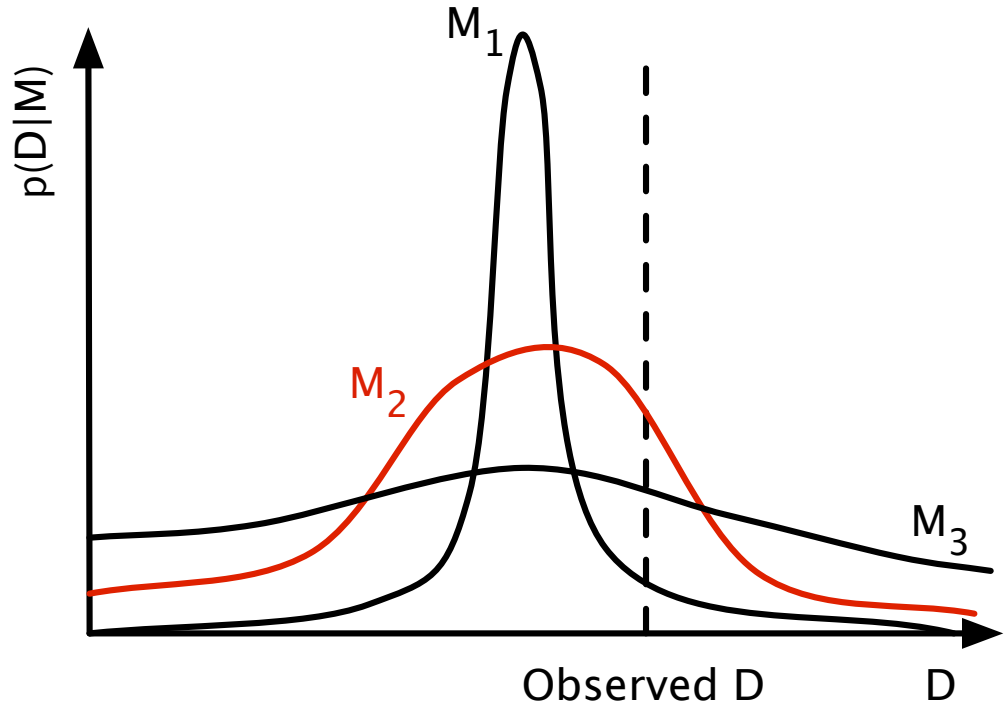
The *evidence* $P(\mathcal{D}|\mathcal{M}_i)$ summarises the information in the data about relative plausibility of models.

Strictly Bayesian approach is then to do *model averaging*, but if computational resources are limited, then we can choose on the basis of posterior distribution.

Evidence and the 'Goldilocks principle' (aka Ockam's razor)

The evidence *automatically* includes a penalty for overly complex models: these can fit a wider variety of datasets (are more flexible), but "spread out" their probability too thinly.

$\mathcal{M}_1$: is too simple.
$\mathcal{M}_3$: too complex.
$\mathcal{M}_2$: 'just right'.



Approximating the evidence

For many models of interest, computing the evidence involves an intractable integral. Hence approximations are needed. Several options:

1. Laplace approximation (Gaussian integral).
2. Bayesian Information Criterion (BIC), application of 1.
3. Variational Bayesian methods.
4. Monte Carlo methods.

We will focus on variational methods, which work by approximating the belief state (distribution over parameters θ).

Yields the **variational free energy**, which can be used as an approximation of $-\log P(\mathcal{D}|\mathcal{M})$.

Outline

Modelling symbolic music

Probabilistic models of symbolic music can, to a large extent, be divided into two broad classes:

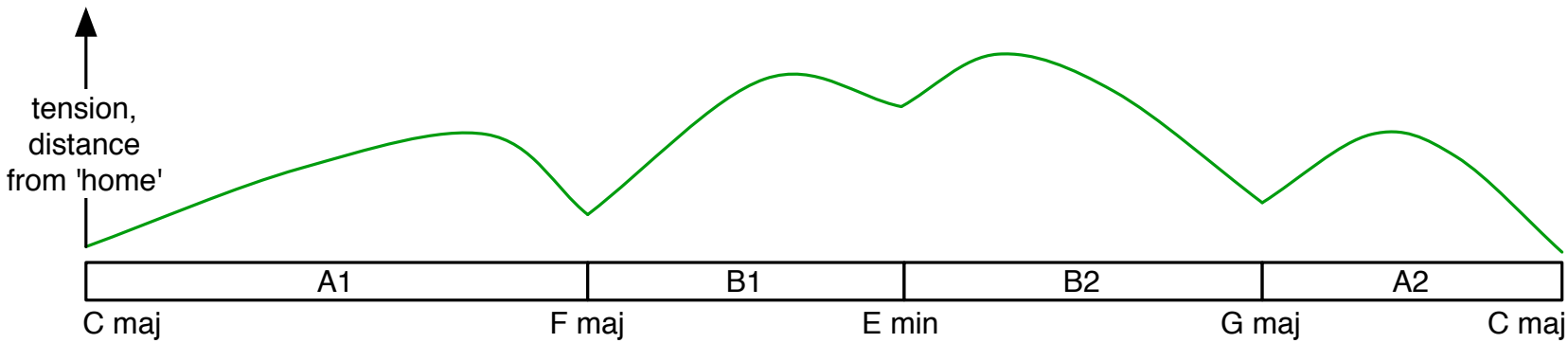
- those based on Markov (or n-gram) models;
- those based on grammars.

Fixed-order Markov 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 [CW95, Pea05] and chord sequences [YG11].

Grammar-based models

They key motivation behind using grammars in music is to account for structure at *multiple time-scales*, which is hard to do with Markov models.

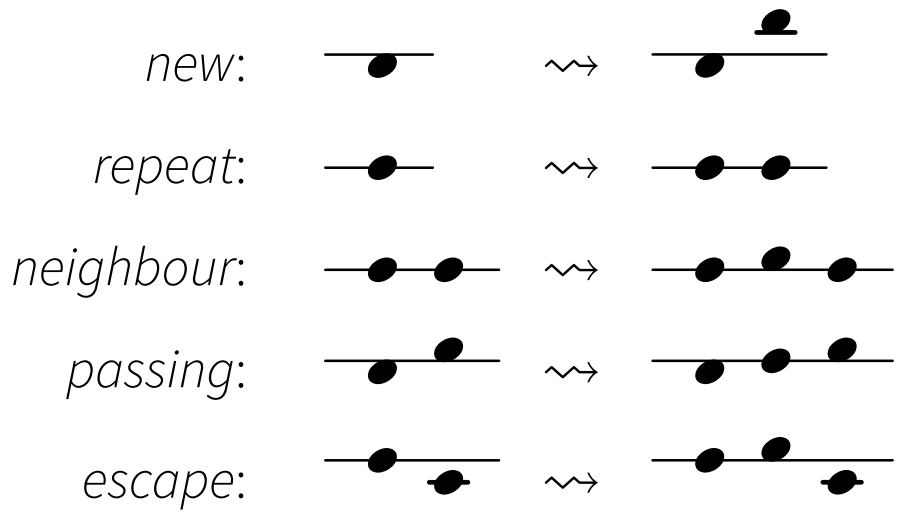


Grammars have been applied in computational musicology since the late 1960s [Win68, Kas67, LS70]. *Probabilistic* grammar-based models of music are a relatively recent development. They can broadly be divided into models of *harmonic* sequence [Roh11, GW13] and models of *melodic* sequence [Bod01, GC07, KJ11]. We will focus on melodic models only.

Gilbert and Conklin's grammar

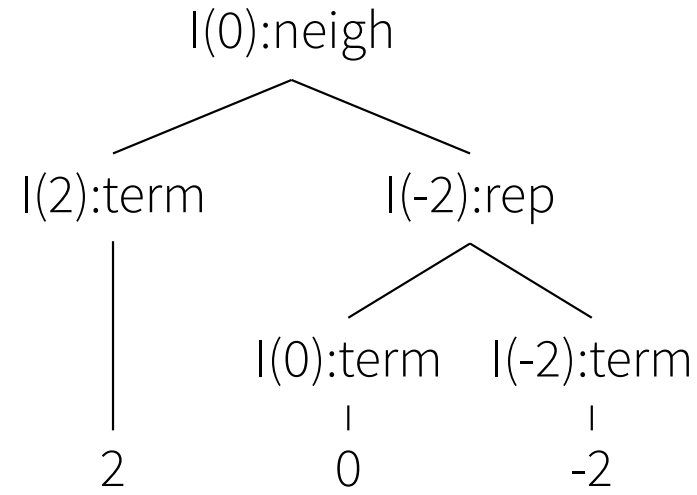
Gilbert and Conklin [GC07] designed a small probabilistic grammar over sequences of *pitch intervals* and proposed that resulting parse trees can be seen as analyses of melodic sequences.

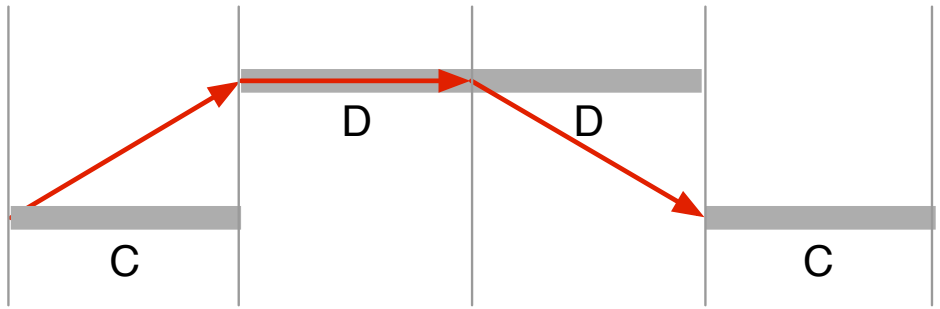
Production rules represent these types of melodic elaboration:



Use of intervals rather than pitches avoids need for context-sensitive rules in the grammar.

Syntax tree over intervals





Markov- *vs* Grammar-based models

Division between ***n***-gram based models and grammar-based models echoes a similar one in computational linguistics, where probabilistic grammars and statistical parsing are used for tasks where a syntactic analysis is required, but ***n***-gram models, especially *variable order Markov models* (e.g. [WAG+09]) are better as probabilistic language models (i.e. they assign higher probabilities to normal sentences).

The situation is less clear in computational musicology—we haven't really got to the stage where we are doing systematic comparisons across a variety of models.

Proposed methodology

This brings us back to our Main Idea:

- Use variational Bayesian methods on a variety of probabilistic models, including probabilistic grammars, to assess and compare models.
- Examine the results of these comparisons to draw musicological conclusions.
- Examine the results of inference on individual pieces to see how well they relate to human perception and analysis.
- Repeat with a variety of musical corpora (e.g. different styles) and again draw musicological conclusions.
- Implement all of this using probabilistic programming languages to provide a uniform environment capable of supporting all sorts of models and automating much of the machinery of learning and inference.

Outline

Probabilistic programming

Probabilistic programming languages aim to provide a powerful environment for defining class probabilistic models, taking advantage of general purpose programming constructs such as recursion, abstraction, and structured data types.

Some are based on logic programming (PHA, PRISM, SLP) while others are based on functional programming (IBAL, Church, Hansei).

We chose PRISM (PRogramming in Statistical Modelling, [SK97]) for this experiment because:

- We get Prolog's DCG notation and meta-programming facilities for implementing our own DCG interpreter.
- We get efficient parsing (like Earley's chart parser) for free, because of tabling in PRISM/B-Prolog.
- We get variational Bayesian learning for free.

A DCG language in PRISM

We designed a DCG language similar to standard Prolog DCGs and wrote a simple interpreter in PRISM. Instead of the usual $Head \longrightarrow Body$ notation, rules are written in one of two forms:

$$Head :: Label \implies Body.$$
$$Head :: Label \implies Guard \mid Body.$$

Guards determine which rules are applicable for a given *Head* term (which may include parameters, as in a Prolog DCG).

Some special DCG goals are:

$+X$:	Produce the terminal X
nil :	Body for an empty production
X~S :	Sample X from PRISM switch S

A PRISM switch name S is a ground term associated with a learnable probability distribution with a Dirichlet prior.

Parameterisation of grammar probabilities

Interpreter creates switches to represent the distributions over clause labels for each non-terminal (rule head).

We implemented two ways of mapping rule heads to switches:

1. **Ground head parameterisation**: Each ground instance of a rule head has its own independent distribution over possible expansions whose *Guard* succeeds.
2. **Head functor parameterisation**: Collects together all rule heads with same functor (name and arity) and the same set of applicable expansions; these share the *same* probability distribution over labels.

The first method is more flexible but tends to result in more parameters.

Parameterisation example

1. **Ground head**

Nonterminal	Switch	Values
$i(\ 0)$	$i(\ 0)$	[*term*, *rep*, *neigh*]
$i(-2)$	$i(-1)$	[*term*, *rep*, *pass*]
$i(\ 2)$	$i(\ 1)$	[*term*, *rep*, *pass*]

2. **Head functor**

Nonterminal	Switch
$i(\ 0)$	*nt*(*i*,1,[*term*,*rep*,*neigh*])
$i(-2)$	*nt*(*i*,1,[*term*,*rep*,*pass*])
$i(\ 2)$	*nt*(*i*,1,[*term*,*rep*,*pass*])

Outline

Corpus

We used the DCG framework to compare the performance of the six models on a corpus of monophonic melodies comprising three datasets (in Humdrum/Kern format):

1. A set of 185 Bach chorales, as used by Gilbert and Conklin.
2. A larger set of 370 Bach chorales.
3. The Essen folk song collection containing 6174 scores.

The full Essen collection was too large to process with the models *gilbert2* and *gilbert3* on our test machine, so two random subsets of 1000 scores each were extracted. These datasets are referred to as: *chorales*, *chorales371*, *essen1000a* and *essen1000b*.

Models overview

1. 0^{th} order Markov model over pitches (*p1gram*).
2. 1^{st} order Markov model over pitches (*p2gram*).
3. 1^{st} order hidden Markov models over pitches (*phmm*).
4. 0^{th} order Markov model over intervals (*i1gram*).
5. 1^{st} 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*).

Markov models over pitches

```
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).
```

PDCG for Markov chains and HMMs over pitch encoded as MIDI note number.

Markov models over pitch intervals

---

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*).

PDCG for 0^{th} and 1^{st} order Markov chains over pitch interval to next note in semitones.

Grammar over intervals (I)

These are the production rules for a grammar based on Gilbert and Conklin's. There are some differences in the *s* rules and the *rep* rule.

% start symbol

s :: *last* $\Longrightarrow$ *i*(*end*).
s :: *grow* $\Longrightarrow$ *P*~*leap*, *i*(*P*), *s*.

i(*P*) :: *term* $\Longrightarrow$ +*P*.
i(*P*) :: *rep* $\Longrightarrow$ *i*(0), *i*(*P*).
i(*P*) :: *neigh* $\Longrightarrow$ *P*=0 | *P1*~*step*, {*P2* **is** −*P1*}, *i*(*P1*), *i*(*P2*).
i(*P*) :: *pass* $\Longrightarrow$ *passable*(*P*) | (*P1*,*P2*)~*passing*(*P*), *i*(*P1*), *i*(*P2*).
i(*P*) :: *esc* $\Longrightarrow$ *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*).

Grammar over intervals (II)

These clauses fix the ranges of the various random switches referred to in the grammar rules.

```
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).
```

Other parameters

The models have “hyper”-parameters to control the prior distributions over the parameters.

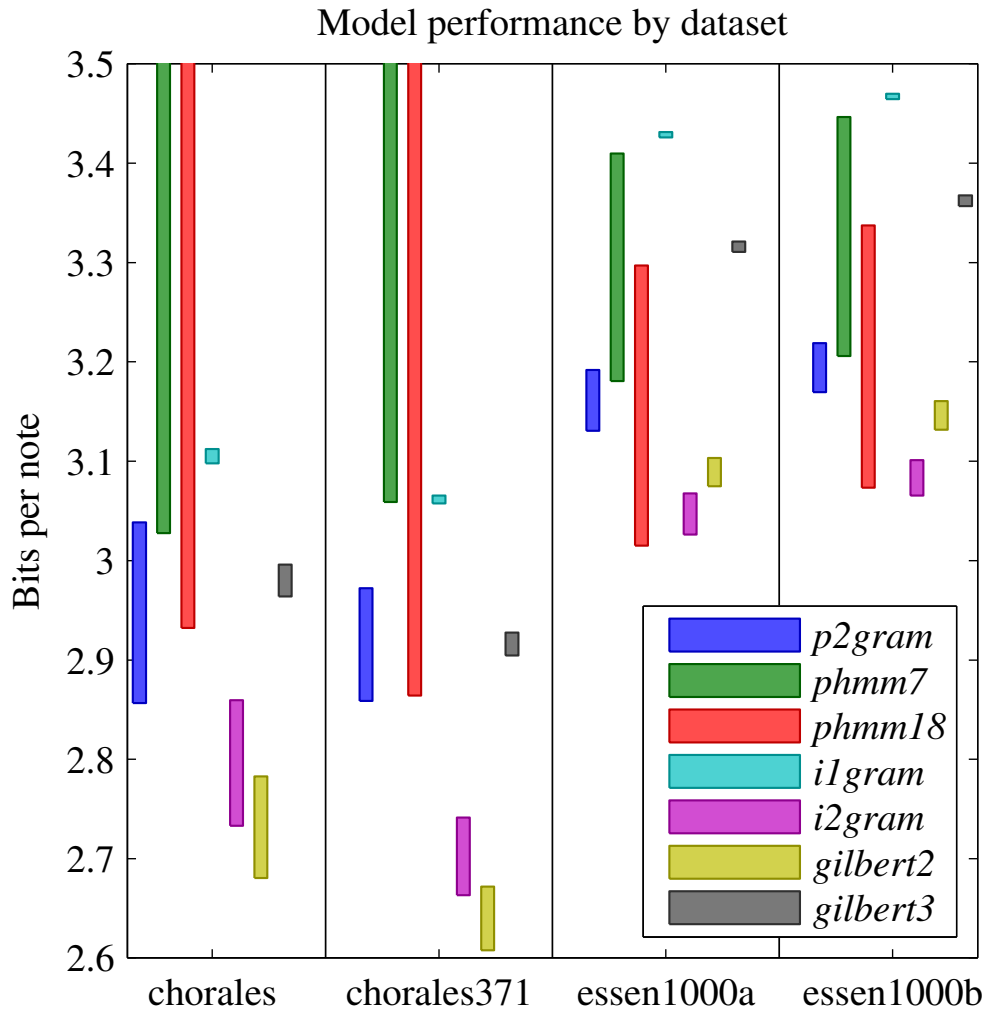
---

prior_weight: {0.3,1,3,10,30}. **% all models**
prior_shape : *shape_spec* **% for markov models**
leap_shape : *shape_spec* **% for grammar models**
pass_shape : *shape_spec* **% for grammar models**
num_states : {1,2,3,5,7,12,18} **% for HMMs**
trans_self : {0,1} **% for HMMs**

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

These affect the initial shapes of distributions over pitches and intervals and *prior_weight* modulates the overall “strength” of the prior, which in turn determines how much data is needed to override prior beliefs.

Results (full dataset)



Overall performance of each model against each dataset.

For each model and dataset, the variational free energy is divided the total number of notes in the dataset to give an idea of the compression acheivable with that model, in bits-per-note or bpn.

The smaller the bpn is, the better the performance.

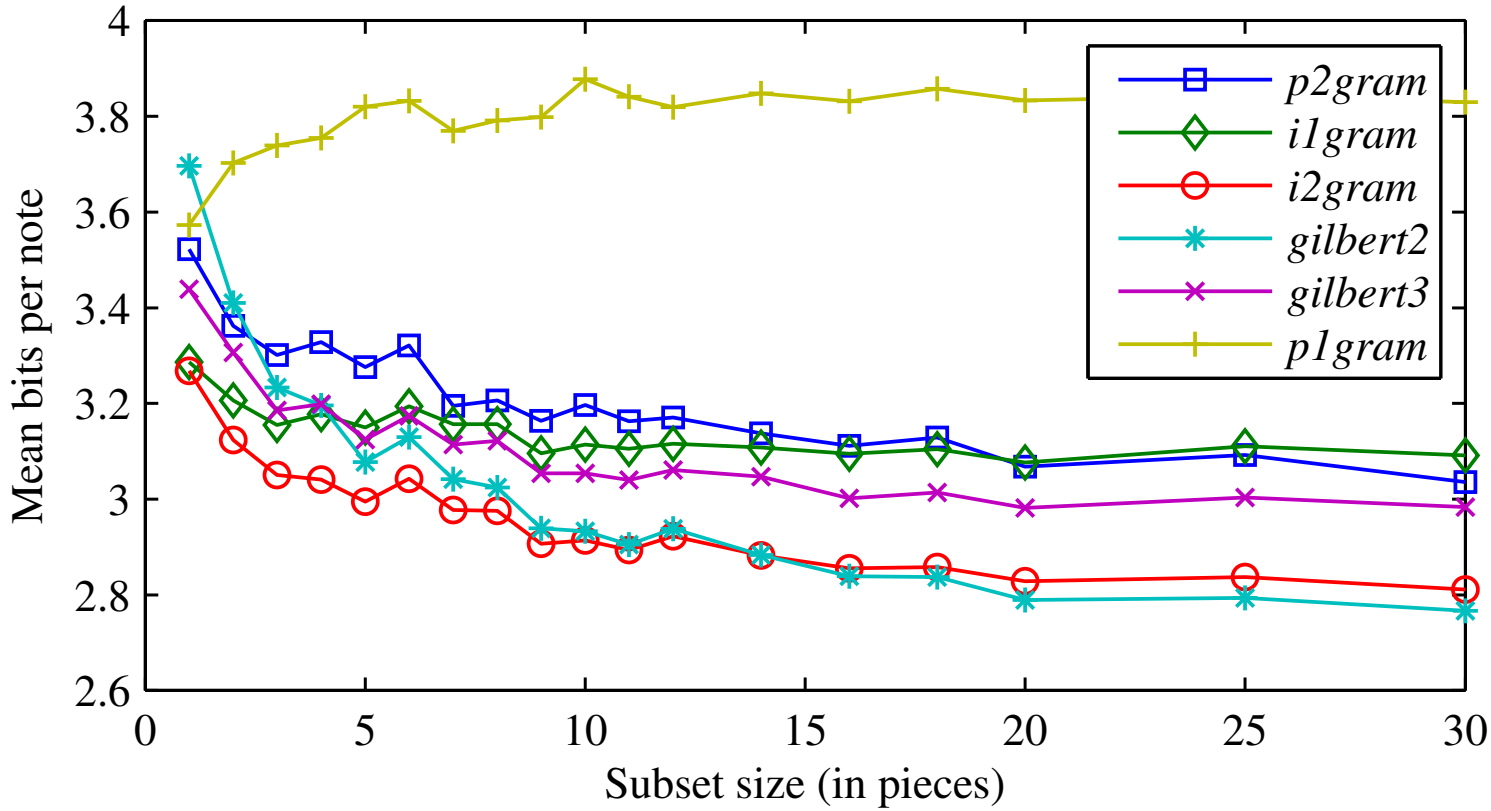
Results (subsets)

In the next few slides, we show how model performance varies with dataset size. These were produced by taking random subsets from the three original datasets. For each subset size K and dataset, we:

1. Choose 20 random subsets of size K from the dataset.
2. Fit all models with all parameter settings to each subset.
3. For each model, choose the parameter settings with best average performance over the 20 subsets.
4. Plot that average performance.

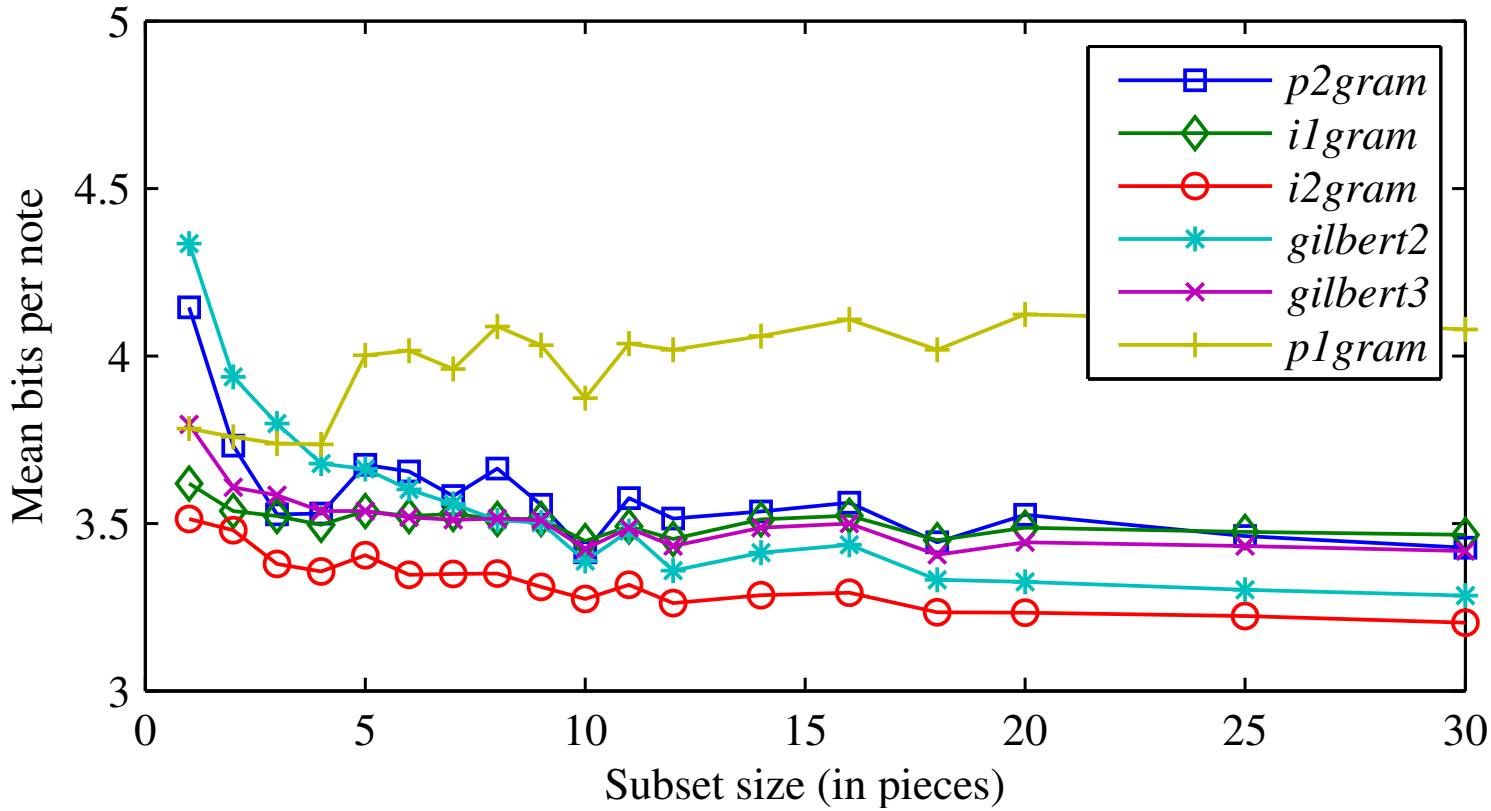
The issue of small dataset performance is relevant to analysing collections of music which are unavoidably small with no hope of finding more examples, e.g. particular (and dead) composers; obscure styles.

Results (*chorales371* subsets)



Subsets results for *chorales371* dataset.

Results (*essen* subsets)



Subsets results for *essen* dataset. Note, *i2gram* is always best.

Conclusions

- Ground head parameterised grammar *gilbert2* is better than the best Markov model (*i2gram*) on Bach chorales.
- Head functor parameterised grammar *gilbert3* is worse than all but 0^{th} order Markov model over absolute pitches (model of marginal pitch distribution).
- 1^{st} order Markov model over intervals out-performed all other models on Essen folk song collection, including tests over all subset sizes. (But HMM with 18 states best case performance was similarly.)
- On Bach chorales, simpler models come into their own with small datasets.

Outline

Conclusions

DCG language was flexible enough to encode a version of Gilbert and Conklin's grammar quite succinctly.

It was encouraging to find that even a relatively simplistic grammar could perform well on the Bach chorales.

However, on the much larger Essen collection, the first-order Markov model over intervals performed best in all cases.

Need to investigate why this is the case. Could be that:

1. The grammar is too simple to capture the structure of the folk songs.
2. The collection is too diverse for a single grammar to cover.

Although the grammar model might seem more sophisticated, designing a grammar that can outperform a Markov model is not at all trivial.

Bibliography I

[Bod01] Rens Bod. Probabilistic grammars for music. In *Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*, 2001.

[CW95] Darrell Conklin and Ian H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[GC07] Édouard Gilbert and Darrell Conklin. A probabilistic context-free grammar for melodic reduction. In *International Workshop on Artificial Intelligence and Music, IJCAI-07, Hyderabad, India*, 2007.

[GW13] Mark Granroth-Wilding. *Harmonic Analysis of Music Using Combinatory Categorial Grammar*. PhD thesis, School of Informatics, University of Edinburgh, 2013.

[Kas67] Michael Kassler. *A trinity of essays*. PhD thesis, Princeton University, 1967.

[KJ11] Phillip B Kirlin and David D Jensen. Probabilistic modeling of hierarchical music analysis. *Analysis*, 1:15, 2011.

Bibliography II

[LS70] Björn Lindblom and Johan Sundberg. *Towards a generative theory of melody*. Department of Phonetics, Institute of Linguistics, University of Stockholm, 1970.

[Pea05] Marcus T. Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, Department of Computing, City University, London, 2005.

[Roh11] Martin Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[SK97] Taisuke Sato and Yoshitaka Kameya. PRISM: a language for symbolic-statistical modeling. In *Proc. 15th Intl. Joint Conf. on Artifical Intelligence (IJCAI)*, volume 2, pages 1330–1335, 1997.

[WAG+09] Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1129–1136. ACM, 2009.

Bibliography III

[Win68] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49, 1968.

[YG11] Kazuyoshi Yoshii and Masataka Goto. A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis. In *ISMIR*, pages 645–650, 2011.