A Critical Look at the Applicability of Markov Logic Networks for Music Signal Analysis

Johan Pauwels, Gyrgy Fazekas, Mark B. Sandler
Centre for Digital Music
Queen Mary University of London
j.pauwels@qmul.ac.uk

Abstract

In recent years, Markov logic networks (MLNs) have been proposed as a potentially useful paradigm for music signal analysis. Because all hidden Markov models can be reformulated as MLNs, the latter can provide an all-encompassing framework that reuses and extends previous work in the field. However, just because it is theoretically possible to reformulate previous work as MLNs, does not mean that it is advantageous. In this paper, we analyse some proposed examples of MLNs for musical analysis and consider their practical disadvantages when compared to formulating the same musical dependence relationships as (dynamic) Bayesian networks. We argue that a number of practical hurdles such as the lack of support for sequences and for arbitrary continuous probability distributions make MLNs less than ideal for the proposed musical applications, both in terms of easy of formulation and computational requirements due to their required inference algorithms. These conclusions are not specific to music, but apply to other fields as well, especially when sequential data with continuous observations is involved. Finally, we show that the ideas underlying the proposed examples can be expressed perfectly well in the more commonly used framework of (dynamic) Bayesian networks.

1 Introduction

Markov logic networks (**?**) form a part of the broader field of statistical relational learning (**?**), which aims to combine complex relational information with probabilistic uncertainty. In the case of MLNs, the relations are expressed through first-order logic (FOL), and the uncertainty as weights assigned to those logic formulas. This way, hard logical rules in a standard FOL knowledge base can be relaxed and an order of importance imposed on the separate logical statements.

One particularly highly structured domain is music. Humans can label music according to a variety of aspects: tempo, harmony, mood, etc. All these labels are strongly related, and if they are time-varying, they are strongly correlated through time too. Trying to exploit those relationships for automatic music labelling is therefore a common research topic. So far, the majority of interdependency models have been constructed using hidden Markov models (HMMs) or other (dynamic) Bayesian networks ([D]BNs) (**?**; **?**; **?**), but Markov logic networks have been proposed (**?**; **?**; **?**) as an alternative framework.

One of the advantages of MLNs is that all HMMs can be reformulated as MLNs. Therefore previous work can easily be reused and be extended upon. However, just because it is theoretically possible to approach a problem as a MLN, does not mean that it is advantageous to do so. Papadopoulos and Tzanetakis illustrate the feasability of MLNs for music signal analysis by listing a number of example systems (**?**; **?**; **?**) that produce the chord label sequence corresponding to a music recording. Yet they do not specifically address why MLNs are preferable to HMMs or, more generally, Bayesian networks. On the other hand, the use of MLNs for those specific examples involves extra complications, which we believe to deserve a more thorough discussion.

In this paper, we try to address these issues, such that a better insight into the advantages and disadvantages of MLNs can be obtained. We look at alternative formulations of some MLN systems, and compare them on their ease of use, so our approach is purely theoretical because the numerical results are expected to be the same.

2 Background

Since Markov logic networks integrate the fields of graphical modelling and first-order logic, a great number of different concepts need to be defined. Furthermore, the application domain of music signal analysis comes with its own vocabulary, combining music theory and signal processing. Due to the limited space available here, and our goal of creating insight into their specific use-cases, we will describe the necessary concepts in an informal, intuitive way with an emphasis on their distinctions. For more thorough and technical definitions, the references should be consulted.

2.1 Music Signal Analysis

The field of music signal analysis aims to give machines the ability to recognise advanced musical concepts in audio recordings. These concepts can be expressed in natural language, such as the mood or genre of a music piece, or can

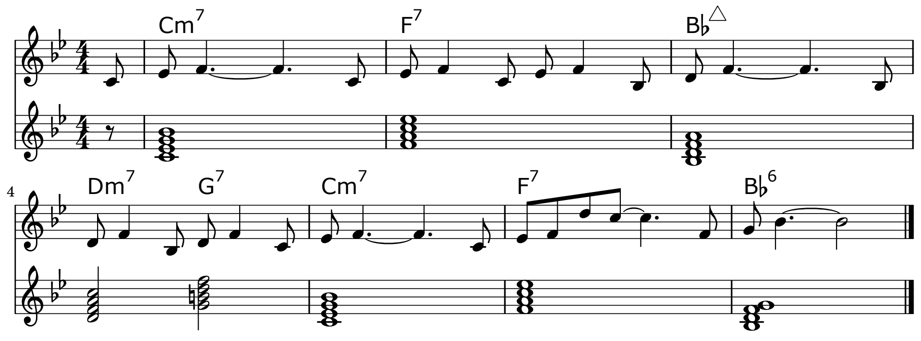


Figure 1: Sheet music example ("Perdido" by Juan Tizol). Chord symbols can be seen above the staves and compactly represent and label the group of notes below them.

be part of the symbolic language known as music notation. Examples of the latter are notes, chords and keys.

The usecase in this paper is the automatic recognition of chords from audio. The easiest way to explain chords is to start from sheet music, of which an example can be seen in Figure 1. Sheet music can be seen a set of instructions for a musician on how to play a piece of music. Since a performance involves a fair amount of human interpretation, many different audio signals can be produced from the same sheet music.

The black stemmed dots in sheet music represent musical notes, displayed on five parallel lines (called a stave). The vertical axis represents pitch height, and the horizontal axis time, read from left to right. The relative duration of the notes is indicated by the shape of their stems. Staves connected by a vertical line at the beginning are played at the same time, and notes that appear in the same horizontal position (in any of the staves) are played concurrently. In our example, most of the time five notes are played together.

Chords are defined by **?** as "the simultaneous sounding of two or more notes". They are represented by the text-like representation above the staves. The complicated part about defining chords is that not all simultaneously played notes contribute to a chord. Some notes can be deemed to belong exclusively to the melody, or not significant enough to contribute to the chord. Deciding which notes get grouped together to form a chord is one of the challenges in chord recognition, and the reason why it is not trivial even when starting from sheet music. Of course, when the input is an audio signal, this complexity gets compounded.

In practice, chord recognition from audio is rarely based on recognising notes (**?**). Instead, chords are directly obtained from a time-frequency representation that is processed with a sliding window. The resulting output is therefore not sheet music with chord symbols as in our example, but a list of text-encoded chord symbols with their associated timings in seconds. In order to obtain temporal consistency over the separately observed sliding windows, probabilistic graphical models are often used to smooth out the windowed output (**?**). Statistical relational frameworks could provide an alternative formulation that has more potential to exploit the strong relationships between different musical concepts (**?**).

2.2 Markov Logic Networks

Probabilistic graphical models are a way to graphically represent (in)dependence relationships between random variables. We can discern two main approaches: Bayesian networks (BN) (**?**, chapter 10) and Markov networks (MN) (**?**, chapter 19). The latter are also known as Markov random fields. Both factorise the joint probability distribution of a set of random variables into a product of more specific distributions. For BNs, these factors are by definition conditional probabilities, whereas for MN the factors can be arbitrary non-negative functions. Because the factors in a BN are conditional probabilities, no normalisation is needed (a product of probabilities is always a probability). In MNs on the other hand, a normalisation term is needed such that all the factors sum to one. Because of this normalisation term over all factors, a MN is not localised and therefore it is harder to perform inference and learning. Its advantage, however, is its greater expressive power.

The difference between BNs and MNs also has consequences for their graphical representations. In both cases, variables are represented by nodes in a network. A BN is represented as a directed acyclic graph (DAG), because conditional probabilities are directional and non-circular. Conditional dependencies are represented by edges from the conditional variables to their dependent variables. A MN is represented as an undirected graph (UG), possibly with loops. Variables that appear in the same factor are densely interconnected by undirected edges. The collection of edges for such densely interconnected nodes is called a clique.

The mapping between factorisation and graph is not univocal though, one graph can represent multiple different factorisations. Furthermore, a graph is more conservative about stating independences than a factorisation: sometimes two variables are independent in the factorisation, even if the graphs says they are not. Never does a graph represent an independence that is not in the factorisation. The case when a factorisation and a graph represent exactly the same set of independences is called a perfect map.

The factors in a MN could be conditional probabilities as well, and in that sense it is trivial to convert a BN into a MN, but there is no guarantee that the associated undirected graph can represent the exact same conditional independences (**?**, chapter 19). Both kinds of graphs have different sets of distributions that they can represent perfectly (i.e. for which they are a perfect map).

Dynamic Bayesian networks (DBNs) are a type of Bayesian network that can represent stochastic processes. Unlike their name suggests, DBNs model stationary processes, the “dynamic” refers to their stochastic nature. Such processes produce sequences of observable and hidden variables, where the sequence index is often a discrete representation of time. Therefore DBNs are naturally represented by chainlike graphs that exhibit a certain regularity in the dependency relationships between their variables. This regularity allows for a compact graphical representation and it is amply exploited by the associated algorithms for inference and learning.

Hidden Markov models[1] (HMMs) (**?**) are closely related to DBNs. The boundary between the two is fuzzy, and is not always consistently defined. We employ the strict definition that an HMM is a DBN with only one (multivariate) hidden variable and one visible variable. If multiple hidden or visible variables arise in the process to be modelled, they are combined to form a single compound variable. This allows to simplify the associated algorithms. The drawback of this approach is that independence relationships within these compound variables are not exploited, whereas DBNs can use them to speed up inference or they can be imposed while learning. A number of specific model architectures branded as HMMs (such as factorial HMMs (**?**), hierarchical HMMs (**?**) and hidden semi-Markov models (**?**)) have been proposed in the literature throughout the years, in order to address these issues. According to our strict definition, we would call them DBNs instead.

Finally, the difference between propositional logic and first-order logic (**?**) is that the former constructs its formulas using true/false statements only, whereas the latter can additionally use quantified variables in its formulas. Through its use of existential ($\exists$) or universal ($\forall$) quantifiers, more generally applicable statements can be made.

3 Unique capabilities of Markov logic networks

One way to look at MLNs, is to see them as an extension of first-order logic with weights. A set of logical statements then no longer complies with a given knowledge base in a binary fashion (yes or no), but an intermediate degree of compliance can be attained. This allows the knowledge base to contain contradictory statements, which will never be fulfilled at the same time.

Thinking about MLNs from a more probabilistic point of view, they can be seen as a way to describe Markov networks (MNs) using first-order logic. One could argue that standard MNs, or more generally probabilistic graphical models (PGMs) (comprising BNs and MNs), already are able to express relationships through the edges in their graphs and they handle uncertainty by associating probabilities to the edges or cliques. The difference is that edges in a PGM can be seen as propositional logic, a simple list of statements that describe which edges are present. If there is a recurring structure in a graph, it would be useful to include this recurrence into the graph description instead of giving an exhaustive list of edges. We could then, for instance, simply state that all nodes of a certain type need to be interconnected. Unfortunately, no mechanism to describe recurrence exists in ordinary PGMs. This is where the added expressiveness of the first-order logic (FOL) in MLNs comes in handy.

To understand the benefits of formulating a graph in FOL, it can be helpful to draw a parallel with the link between Bayesian networks (BNs) and dynamic Bayesian networks (DBNs). Any given DBN does not model its underlying stochastic process for one specific sequence length only, but for any sequence length. If we want to model the same process using an ordinary BN, we would need to create multiple networks, one for each of the process lengths under consideration. In other words, a DBN adapts seamlessly to the length of observed sequences that are presented to it. The advantage of DBNs over ordinary BNs is therefore twofold. First, it allows a compact and elegant formulation of a whole set of BNs that exhibit a specific regularity through time. Second, that regularity can be exploited by inference and learning algorithms that are more optimal than the algorithms used for general BNs. In contrast, the drawback of a DBN is that it can only describe a limited subset of the number of possible BNs: only those that are created through repeating a set of nodes as many times as necessary, a process known as *unrolling* the network. The descriptive power of a DBN is therefore higher than propositional logic. It can be considered as a small subset of first-order logic, where only the sequence index variable can be universally quantified ($\forall$).

The advantages of MLNs over MNs are similar to the advantages of DBNs over BNs: both provide a convenient way to describe some regularity in a network, such that the formulation is suitable for observations of varying dimensions and such that the regularity can be exploited by the algorithms. MLNs are more broadly applicable than DBNs though: any regularity in an arbitrary number of dimensions can be described because the full descriptive power of first-order logic can be used. Any MN can therefore be described as an MLN: in case no regularity is present, the description will only use propositional logic. Obviously, the advantage of using MLNs is only noticeable when describing systems that cannot easily be described with propositional logic.

Multiple toolkits for MLNs are available that turn the underlying ideas into useable software. Among them is the reference implementation developed by the creators of the MLN theory, called Alchemy (version 2.0)[2]. Its latest update stems from January 2013, however. A more recent toolkit is named ProbCog[3] (last update July 2019), originating at the Technical University of Munich. Different toolkits implement different logic engines though, which means that their syntax and capabilities can slightly differ. Since the example MLNs given by Papadopoulos and Tzanetakis (**?**;

[1]The naming of HMMs is rather unfortunate in the sense that they are not Markov networks, but Bayesian networks.

[2]https://alchemy.cs.washington.edu/

[3]https://github.com/opcode81/ProbCog

?) are solved by the ProbCog toolkit, we will consider that variant in the remainder of this paper.

One remark is that research on MLN first focussed on completing the functionality for describing a MN through logic. The potential advantage of improving the inference and learning algorithms – by exploiting the regularity present in the logic description – is much more of a work-in-progress. A useful fallback option is to instantiate the underlying graphical model first in a process called grounding (comparable to the unrolling of a DBN into a BN by copy-pasting the repeated part for each sequence step) and then reuse the existing algorithms for standard Markov networks. Performing inference of MLNs without the need to ground the network first, is called *lifted inference* and is the subject of ongoing research (**?**; **?**; **?**).

Some of the first lifted inference algorithms are available in the Alchemy software, but the ProbCog toolkit only contains algorithms that work on grounded networks. Papadopoulos and Tzanetakis advocate the use of toulbar2 [4], an exact solver for Markov networks and other cost function networks included with ProbCog. The drawback is that one of the theoretical advantages of MLN over MN disappears in practice. This leaves us with the compact and elegant notation as main benefit of using MLNs. In the following sections, we will discuss why the particular requirements of music signal analysis make MLN formulations less elegant than they would be for other application domains.

4 Disadvantages of MLNs as implemented in the ProbCog toolkit

In a typical music signal analysis task, the goal is to take an audio signal as input and return one or more labels that describe the content of the music. In the example systems given by Papadopoulos and Tzanetakis, these labels are chords, a time-varying description of harmony. It is therefore natural to model them as a chainlike graph generated by a stationary process. The lack of explicit support for sequences in ProbCog is therefore unfortunate and forces us to use less elegant workarounds. Another typical characteristic of music analysis systems is that the observations are generally continuous (because they are in one way or another derived from a continuous waveform). Specifically for chord analysis, the observations are often a 12-dimensional time-frequency representation where frequencies are reduced to a single octave, known as a chromagram (**?**; **?**). As as result, the fact that ProbCog does not support continuous probability distributions is another drawback that requires a workaround.

4.1 Handling continuous observation probability distributions

The solution to handle continuous probability distributions proposed by Papadopoulos and Tzanetakis is to precalculate the probabilities of all observations in a specific song (as opposed to specifying the probabilities for all possible observations as a discrete observation distribution). The set of observations per song is then used as the discrete domain and each "discrete value" is encountered exactly once. The weighted formulas (notated as a numerical weight separated by whitespace from a logic formula) they propose for modelling the observations have the form

$$\text{weight}\left(\mathit{Label}, N\right) \qquad \text{observation}\left(\mathit{Obs}_N, t\right) \wedge \text{chord}\left(\mathit{Label}, t\right) \quad (1)$$

with Obs_N the name given to the observations at index N. We use the convention that logical variables are represented by lowercase text (such as t) and logical constants by capitalised text (such as Label). In practice, such a formula needs to be specified for every combination of the label and observation constants because the weights are dependent on both. The logical constants then take values such as *Cmaj* and *Amin* for Label and $0, 1, \ldots$ for N. Papadopoulos and Tzanetakis employ 24 chord labels and with a typical song being divided into hundreds of beats (time segments), the number of formulas that needs to be specified runs into the thousands.

The corresponding evidence needs to be provided for every song in the form of

$$\text{observation}\left(\mathit{Obs}_N, N\right) \quad (2)$$

which effectively says that "the observation at time N is called the observation at time N". The whole process entails the creation of a custom MLN per audio file under analysis, instead of defining one model that can adapt to any sequence of observations, as would be the case for true discrete observations.

Because each of the discretised observations is guaranteed to appear only once in the sequence, the validity of the FOL observation formulas is limited to singleton domains. It is superfluous to specifically create named constants Obs_N just to link the time sequence index t to the index of the discrete set of observations n (they are always equal). The combination of formulas of the form (1) and evidence of the form (2) can be simplified by substituting the sequence index variable t by the only valid index N in the observation predicate. The resulting formulas are then of the form

$$\text{weight}\left(\mathit{Label}, N\right) \qquad \text{observation}\left(N\right) \wedge \text{chord}\left(\mathit{Label}, t\right) \quad (3)$$

with evidence of the form

$$\text{observation}\left(N\right) \quad (4)$$

Now the sole reason of existence for the observation predicate is to pass on the index N to the variable t in the chord predicate, so we can remove the former altogether by substituting t by N again. The final formula is then

$$\text{weight}\left(\mathit{Label}, N\right) \qquad \text{chord}\left(\mathit{Label}, N\right) \quad (5)$$

with no need to specify evidence. This formulation is not only much simpler than the originally proposed one, it also is in propositional logic form, showing that the added complexity of the first-order logic formulation is unnecessary in this case, and only obfuscates the formulation.

In summary, using continuous observation variables demands that the observation probabilities are calculated externally and that they are written in time-unrolled (propositional) form into the MLN configuration on a file-per-file

[4]http://www.inra.fr/mia/T/toulbar2

basis. Consequently, the separation between logical network definition and evidence is compromised and most of the elegance of the logical description gets lost. The actual file-dependent observations get absorbed into the model definition instead of being evidence. Looking beyond the ProbCog toolkit, some work has been done to extend the Alchemy MLN toolkit with continuous observation distributions (**?**), but those distributions are limited to Gaussians. Since the observation probabilities in (**?**; **?**; **?**) are calculated as the correlation between chroma vectors and theoretical templates, using Alchemy is not a viable solution.

4.2 Handling sequences

The underlying problem that makes sequences hard to handle in ProbCog, is that there is no support for arithmetic operators between variables. This absence becomes especially notable when we want to encode the probabilities of pairwise transitioning between chords (such as in an HMM) into formulas. We want to specify that the probabilities only apply to two timestamps t_1 and t_2 if they are related as $t_2 = t_1 + 1$. The way this problem is solved in (**?**; **?**; **?**), is by adding an extra helper predicate (here called $\mathrm{next}\,/2$) to the logic formula to control when the formula is valid:

$$\begin{aligned}&\mathrm{weight}\left(Label_1, Label_2\right)\\ &\mathrm{chord}\left(Label_1, t_1\right) \wedge \mathrm{chord}\left(Label_2, t_2\right) \wedge \mathrm{next}\left(t_1, t_2\right)\end{aligned} \tag{6}$$

It is then necessary to provide evidence of the form $\mathrm{next}(N, N+1)$ as many times as necessary for the song length. So just like for the observation formulas, this requires manual unrolling of time.

Looking at it graphically, this means that the (hidden) chord nodes of all sequence indices in the model are fully interconnected. In order to "activate" the connection however, the associated predicate $\mathrm{next}\left(N_1, N_2\right)$ between a specific pair of timestamps $\left(N_1, N_2\right)$ needs to be given as evidence[5]. Only in that case does the size three clique formed by formula (6) become true. The model definition itself thus leaves the possibility of non-linear time open, e.g. 4–4–2–6–5–1–4, and it relies on the given evidence to make it linear, e.g. 1–2–3–4–5–6. Again, we see that the separation between model definition and evidence gets blurred, because the model relies on specific evidence to be given in order to complete the network definition.

Furthermore, the $\mathrm{next}\,/2$ predicate cannot be reused to make additional, arbitrary connections between non-consecutive chord nodes (for instance to probabilistically tie repetitions of the same chord subsequence together, as proposed in (**?**)). Specifying $\mathrm{next}\left(t_1, t_3\right), \forall t_3 \neq t_1 + 1$ as evidence does create such an additional connection, but its associated weight would still encode the pairwise probability between consecutive chords, leading to unintended and incorrect transition probabilities. Any additional connection then requires the introduction of another $\mathrm{next}\,/2$-like predicate, connecting all chord node combinations together once more in supplementary cliques of 3. The number of nodes and connections in the network rapidly soars this way.

In short, the absence of arithmetic operators in ProbCog makes a FOL network description instantiate suboptimal graphs. The result is fully connected and relies on the presence of evidence to prune the nonsensical connections. This approach is way more convoluted than defining only those connections that need to be made in the first place. An alternative to accomplish the latter, would be to get rid of the $\mathrm{next}\,/2$ predicate and write formulas down in propositional logic form only when a connection between the two sequence indices T_1 and T_2 is actually required. The form will then be

$$\begin{aligned}&\mathrm{weight}\left(Label_1, Label_2\right)\\ &\qquad\mathrm{chord}\left(Label_1, T_1\right) \wedge \mathrm{chord}\left(Label_2, T_2\right)\end{aligned} \tag{7}$$

without any corresponding evidence. The drawback of specifying the transition probabilities through propositional logic, is that the number of formulas for the model configuration will increase. But because the configuration needs to be song-dependent and time-unrolled for previously mentioned reasons, the configuration file realistically needs to be generated programmatically anyway. Therefore, this is not an issue in practice.

Alchemy does have partial support for arithmetic operators, but our desired use-case – in which the transition probabilities would be elegantly formulated in the FOL form below – is not supported.

$$\begin{aligned}&\mathrm{weight}\left(Label_1, Label_2\right)\\ &\qquad\mathrm{chord}\left(Label_1, t\right) \wedge \mathrm{chord}\left(Label_2, t+1\right)\end{aligned} \tag{8}$$

In addition, Alchemy does not include an exact inference algorithm.

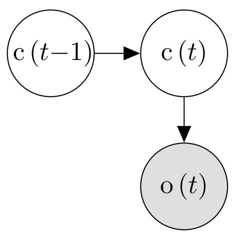
5 Alternative formulations of concrete MLN examples

The difference between the MLN systems proposed in (**?**; **?**; **?**) and other work on automatic chord estimation is twofold: the former can use the full expressive power of first-order logic to describe relations instead of the more limited expressiveness of the latter, and the underlying graphical model is a Markov network instead of a Bayesian network. We have shown in the previous section that in practice, the full power of FOL cannot be used when describing chord estimation systems. In this section, we will reformulate the concrete MLN examples as (dynamic) Bayesian networks to demonstrate that their underlying ideas can be expressed in this more conventional framework as well. It also makes it easier to compare the proposed approaches to other systems, for example to previous work integrating key and chord estimation (**?**; **?**).

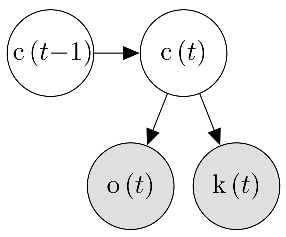
Common to all systems in (**?**; **?**; **?**) is that they focus on inference. The weights are set manually instead of being learned. We can divide the examples into two groups. The first contains the three systems first described in (**?**) and later repeated in (**?**). The first system estimates chords based on chroma observations, the second additionally takes prior key

[5]We assume a closed world, such that withholding evidence is the same as giving evidence of its negation

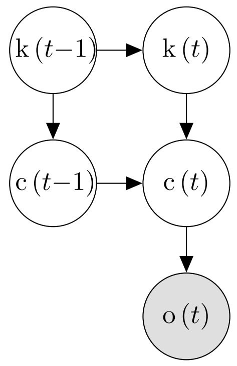
information into account and the third performs joint key and chord estimation. All three systems have weights that are derived from conditional probabilities and only consider pairwise time dependencies. Added to the fact that all formulas in the MLNs are conjunctions of positive literals, this means that the resulting MNs instantiated from the MLNs can be equally well represented as DBNs (**?**).



(a) chord estimation



(b) chord estimation with prior key

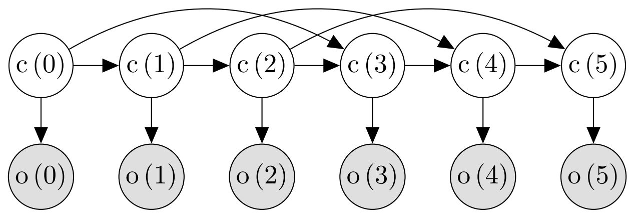


(c) joint key and chord estimation

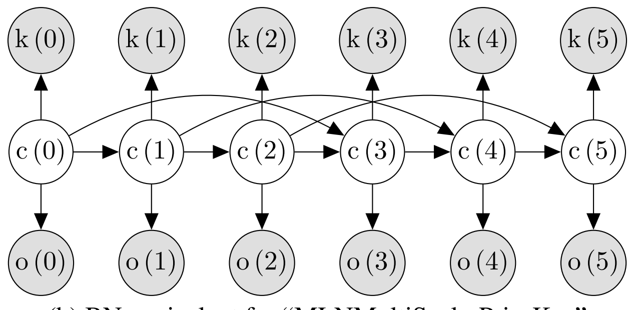
Figure 2: Equivalent DBN representations of the systems presented in (**?**), where $\mathrm{o}\,(t)$, $\mathrm{c}\,(t)$ and $\mathrm{k}\,(t)$ stand for respectively the chroma observation, chord and key at time t. Shaded nodes represent observed variables and white nodes are hidden variables.

The diagrams for those three systems are displayed in Figure 2. The formulations as DBNs will actually be more compact and elegant because continuous observations densities can be used and the observation probabilities no longer need to be unrolled in time. Moreover, the DBN can be reused multiple times by changing the observations presented to it, whereas the MLNs need to be redefined for every song because of the conflation between model and evidence. Another advantage of this formulation is that optimised inference algorithms can be used. For instance, we can now see that it is not surprising that Papadopoulos and Tzanetakis found the difference between the chord estimation MLN and a chord HMM to be statistically not significant (**?**), as they describe exactly the same system. The only difference is that the MLN description needs to resort to a more generally applicable inference algorithm because of the time-unrolled observations and suboptimal network layout, whereas the HMM can use the Viterbi algorithm for decoding (resulting in a speedup of orders of magnitude).

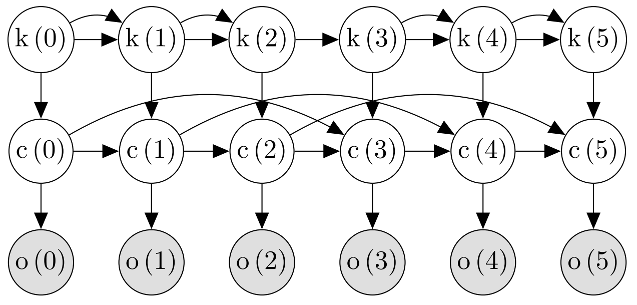
The second group of MLN examples is arguably more interesting, precisely because they cannot be formulated as DBNs (but still as BNs). It comprises three of the systems presented in (**?**), one of which was previously presented in (**?**). The motivation behind them is the fact that one drawback of DBNs is their inability to take longer term dependencies between chords directly into account. We assume that we dispose of a number of pairs that indicate high similarity between its (non-adjacent) constituents. These similarity pairs can come from a prior structural analysis under the assumption that a repetition of a chorus, for instance, will contain the same chords as the previous chorus. Papadopoulos and Tzanetakis use manual structural annotations to derive these similarity pairs, such that they are binary (either similar or not). The repetitions are then exploited to improve the chord estimation. In this sense, this approach is comparable to other systems that take into account repetition, such



(a) BN equivalent for "MLNStruct"



(b) BN equivalent for "MLNMultiScale-PriorKey"



(c) BN equivalent for "MLNMultiScale"

Figure 3: Equivalent BN representations of the more complex systems presented in (**?**), where $\mathrm{o}\,(t)$, $\mathrm{c}\,(t)$ and $\mathrm{k}\,(t)$ stand for respectively the chroma observation, chord and key at time t. Shaded nodes represent observed variables and white nodes are hidden variables. It is assumed that nodes $[0:2]$ and $[3:5]$ belong to the same measure and that the time segment pairs $(0,3)$,$(1,4)$ and $(2,5)$ are given as highly similar.

as (?; ?). The difference is that here the knowledge about the repetitions is integrated probabilistically into the inference, whereas those previous approaches do a deterministic early fusion of the repeated observations.

In essence, the three systems are constructed by unrolling the previous three DBN systems through time such that the similarity knowledge can be integrated by drawing additional edges between the similar time segments. Example graphs with six time segments are shown in Figure 3. The most complex example not only takes long term chord similarity based on structure into account, but also medium term key similarity by connecting keys in the same measure under the assumption that key changes within a measure are highly unlikely.

Inference in the resulting BNs will not be faster than in the optimal, propositionally described MNs (which have the same graphs, but with undirected edges instead), but the BNs have the advantage of being easier to interpret. The weights proposed in (?) for the similarity-based edges are fixed without justification or interpretation, presumably found through a parameter sweep. Because of the global normalisation in MN, this makes it hard to interpret the numbers. Even worse, the interpretation of the other weights (derived from conditional probabilities) will be affected by the addition of the supplemental structural edges, which are by definition unpredictable and song-dependent, and depend on the given evidence (?). The whole graph then quickly becomes opaque. In a BN, the structural edges will get a conditional probability assigned, which has to be normalised locally to preserve the total probability in each node. The optimal parameter still can – and likely has to – be found through a parameter sweep, but the result will be interpretable and will not affect the other parameters in unforeseeable ways. A BN will also make it easier, for example, to consolidate the key transition probabilities and the probabilistic tying of keys within the same measure in the last graph of Figure 3 into a single edge (currently they are represented by separate edges connecting the same nodes).

6 Conclusion and future work

In this paper, we studied MLNs for music analysis by analysing a couple of previously proposed example systems for chord estimation and reformulating them as (dynamic) Bayesian networks. The theoretical advantages of MLN are an easier, more powerful formulation of graphical models and an improvement in learning/inference algorithms through exploiting regularity in the networks. In practice, however, we saw that the MLN systems formulated in (?; ?; ?) are not optimally exploiting the capabilities of the MLN framework. This is partly because these formulations introduce unnecessary variables to create FOL where propositional logic would suffice, and partly because the lack of support for continuous observation distributions and sequences forces the use of convoluted workarounds. We furthermore showed that it is not necessary to resort to Markov networks to express the underlying ideas, but that Bayesian networks can use the same principles, with improved interpretability. We argue that because music signals are variable through time and causal, a directional left-to-right relationship arises naturally when modelling music,therefore the associated graphs will be chainlike. These conclusions are not specific to music, but apply to other fields as well, especially when sequential data with continuous observations is involved.

This is not to say that MLNs are not useful for solving a number of problems, even in the musical domain. Candidate problems for which MLNs would provide a good fit have only discrete variables and would ideally model static processes. Any data that is topographically organised would be suitable, as such a layout increases the chance of cyclical relationships and regularity that cannot be properly modelled by (dynamic) Bayesian networks. In the musical domain, one application that fulfils these requirements would be similarity discovery or clustering between artists, as long as no time-varying features are used.

Another case where MLNs can be helpful is when a large body of knowledge about the problem at hand already exists in first-order logic form. In contrast, the formulas in (?; ?; ?) are relatively simple, just describing relationships between node pairs. In this scenario, the related paradigm of Bayesian logic networks (BLNs) (?) might be an alternative if no bidirectional connections are needed. An implementation is available in the same ProbCog toolkit, which has the advantage that arithmetic are supported such that it is possible to impose $t_2 = t_1 + 1$ through a logical rule. Continuous observation distributions require the same workaround as with MLNs though.

Another alternative for MLNs we could explore in the future is ProbLog (?). The advantage of this approach is that it is built around a complete Prolog engine, which makes it easy to extend with custom functionality, such as arbitrary continuous distributions. In our first preliminary experiments, however, we already encountered problems to make it scale to the typical network sizes of our problems, despite recent advances in this aspect (?).

Acknowledgements

This work has been partly funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1 and by the European Unions Horizon 2020 research and innovation programme under grant agreement N° 688382.

Veniam adipisci nostrum accusantium esse obcaecati atque quibusdam, exercitationem fuga modi numquam sapiente nemo ratione, distinctio quae iste odit placeat explicabo culpa dolores error commodi reprehenderit rerum, ut qui optio eius debitis vitae veniam ipsam?Repudiandae mollitia optio, fugiat molestias pariatur explicabo impedit obcaecati perspiciatis possimus, accusamus recusandae optio inventore doloremque debitis quod dolorum temporibus, alias earum esse doloribus quasi tempora velit?Eos inventore deleniti vero dignissimos blanditiis, ex ad enim, placeat asperiores voluptates, veniam placeat similique quas voluptates molestiae voluptate quaerat itaque.Ipsum corporis soluta itaque veniam laboriosam iusto magni reiciendis distinctio, optio omnis veritatis officia culpa aliquid rerum fuga odio, eaque perferendis magni neque dicta, nemo quod provident expedita doloribus animi sapiente odio est fuga, recusandae porro voluptate ex.