DRUM'N'BAYES: ON-LINE VARIATIONAL INFERENCE FOR BEAT TRACKING AND RHYTHM RECOGNITION

Charles Fox, Iead Rezek, Stephen Roberts
Department of Engineering Science
University of Oxford

ABSTRACT

It is useful for music perception and automated accompaniment systems to perceive a music stream as a series of bars containing beats. We present a proof-of-concept implementation of a Variational Bayesian (VB) system for simultaneous beat tracking and rhythm pattern recognition in the domain of *semi-improvised* music. This is music which consists mostly of known bar-long rhythm patterns in an improvised order, and with occasional unknown patterns. We assume that a lower-level component is available to detect and classify onsets. The system uses Bayesian network fragments representing individual bars and infers beat positions within them. Model posteriors provide principled model competition, and the system may be seen providing a Bayesian rationale for agent-based and blackboard systems. The psychological notion of priming is used to instantiate new candidate models.

1. INTRODUCTION

Perception of beats and bars is difficult because there is scope for much ambiguity. Fig. 1(a) shows an example of *between-bar* ambiguity: a listener familiar with military bands would probably chunk this into 'marching' bars beginning at the points marked '*', but a listener more accustomed to electronic dance music might chunk it into classic 'two-step' drum'n'bass bars beginning at the '†' points. *Within-bar* ambiguity is illustrated in the two bars of fig. 1(b): here the rows represent different onset types; the first bar is a standard 4/4 pattern with the snare on beat 3. The second bar is similar but the snare hit occurs later. This could be interpreted in at least three ways: (1) the pattern could be the same standard 4/4 as in bar one, but with an onset-detection error giving rise to the snare being perceived as late; (2) the pattern may be the same standard 4/4 but the player made a (possibly deliberate) mistake of playing the snare late; (3) it may be a new pattern with the snare on beat 3.5. The difference between (2) and (3) is subtle and even debatable: if the player makes the same 'error' of (2) several times then we begin to perceive it as a new pattern as in (3).

Both types of ambiguity are usually resolved by prior information. The genre of the performance favours certain patterns: a composition is unlikely to switch from military march to drum'n'bass in consecutive bars. We expect – very strongly – that each new pattern will begin at exactly the time that the previous pattern ends, and – less strongly – that it will have a similar tempo.

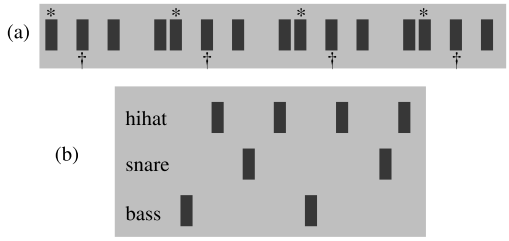

Figure 1. (a) Between-bar and (b) within-bar ambiguity.

We formalize these ideas using Bayesian network models of single-bar rhythms to infer *within*-bar beat positions, together with a blackboard framework to select *between* these models. The required networks are highly loopy so exact inference is intractable. We have selected the Variational Bayes inference approximation because it seeks an estimate of the joint rather than the marginals, and we are more interested in coherent explanations of whole bars than of individual notes. VB is fast enough to run in real-time, unlike sampling methods which may be very time consuming. VB runs iteratively, so may be terminated early to read off a "best decision so far" which is useful in the real-time setting. Finally, we need to compare rival explanations of bars and VB can provide a computationally cheap lower bound on the model likelihood using byproducts of its parameter computations.

VB is now a standard machine learning technique and we present its application to a novel musical task. We implement the message passing VB computation scheme of [5] and extend it by introducing annealing and undirected sibling rivalry links. We introduce the problem of assigning data points to whole models, and heuristics of priming and pruning for dealing with this.

Real-time music perception systems generally use one of two architectures. *Dynamic state* models (e.g. [2] and the Hidden Markov Model component of [3]) update the state of the music at each point in time (typically each frame of audio). In contrast, *position-in-time* models (e.g. [1] and the Bayesian Network component of [3]) maintain beliefs about the positions of a set of musical events – such as note onset or beat times. Our approach is a position-in-time method, but unlike these previous systems it deals with *semi-improvised* music: rather than require the per-

formance to be aligned with a fixed score, we consider sequences of known (and occasionally unknown) drum patterns played in an improvised sequence. Within each pattern, notes may also be omitted from or inserted into the standard pattern. In particular this means that unlike previous systems, we cannot assume a given mapping from input notes to model notes. We consider multiple rhythm models as hypotheses to explain the data: when the models are viewed as 'agents' this is a similar approach to non-probabilistic agent-based methods such as Goto [6] and Reis [8]. However our models have rigourous probabilistic semantics, for their internal parameters – in the manner of Raphael [3] and Cemgil [1] – but also for their model posteriors. It is the latter that provides a rigourous probabilistic basis for agent competition.

We assume that some lower-level system is available (e.g. [4]) which reports incoming onsets with a belief over their time of occurrence and a discrete classification between bass, snare and hihat type hits. These could be extracted from drum audio or from other rhythmic instruments such as guitar chord strums.

2. VARIATIONAL INFERENCE

Once loops are present in a Bayesian network model M, exact inference of the data-conditioned joint $P(x_1, ..., x_N|d)$ and its Maximum A Posteriori (MAP) solution become intractable and approximations must be used. The VB approximation assumes a mean-field factorization:

$$P(x_1, ..., x_N|d, M) \approx Q(x_1, ..., x_N) = \prod_i^N Q_i(x_i)$$

where the node distributions $Q_i(x_i)$ are of the same form as the transition probabilities $P(x_i|par(x_i))$ and $par(x_i)$ are the parents of x_i. VB seeks parameters of the Q_i so as to minimize the KL divergence $KL[Q||P]$. It can be shown (e.g. [5]) that this may be accomplished by iteratively updating individual nodes distributions with the VB update rule

$$Q_i(x_i) \leftarrow \langle \log P(x_i|mb(x_i)) \rangle_{Q(mb(x_i))}$$

where mb denotes the Markov Blanket of the node. When nodes are conjugate exponential the expectation has an analytic solution and the update is computationally simple. As shown in [5], the model log likelihood, $\log P(d|M)$, has a tractable lower bound:

$$\sum_{i=1}^{N} \left[\langle \log P(x_i|par(x_i)) \rangle_Q - \langle \log Q_i(x_i) \rangle_Q \right].$$

3. SINGLE MODELS

We model the whole performance as a series of Bayesian networks, stitched together. Each network models a known 4/4 single-bar rhythmic pattern. At each bar, we compare several models to find the best fitting one, and the MAP

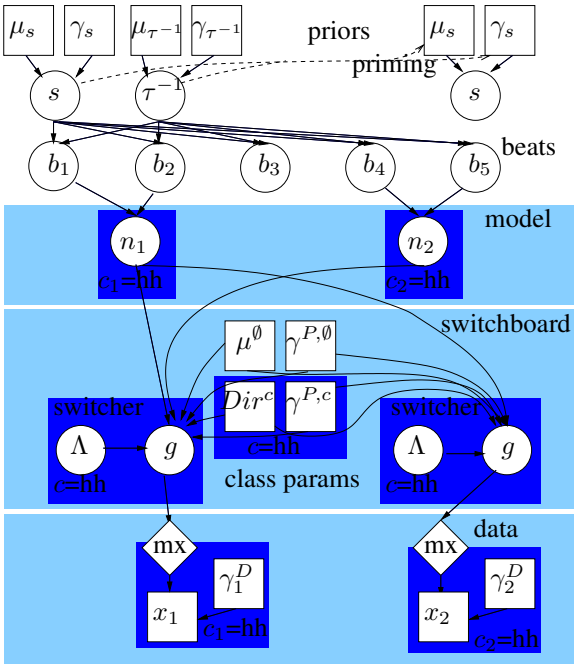

Figure 2. Schematic of a single-bar Bayesian network.

values of its parameters. These parameters may then be used to extract the model's estimate of the past, present and future beat locations as required. (We leave aside the question of how and when to instantiate models, but will return to it in section 4.)

Fig. 2 shows a model for a very simple rhythm consisting of just two hihat (hh) notes. Fixed nodes are shown as squares and hidden nodes are circles. The diamond-shaped *multiplexer* nodes will be discussed later: for now, we may assume they simply forward incoming and outgoing messages unchanged.

The goal is to infer the beat positions $\{b_i\}_{i=1:5}$, along with the start time s and tempo τ. To utilize the efficiency of conjugate-exponential models, we work with the inverse tempo, τ^{-1} instead of τ. Both s and τ^{-1} are modelled as Gaussians, with fixed mean and precision priors.

The beat positions are modelled as linear combinations, $b_i = s + i\tau^{-1}$, and note positions are modelled as the weighted linear combination of their two nearest beats, $n_j = b_{i(j)} + w_j(b_{i(j)+1} - b_{i(j)})$ where $i(j)$ is now a function specifying the latest beat number i before the jth note, and w_j is a rational weight with $0 < w_j < 1$.

Incoming observations are shown in the lowest layer of fig. 2 . We assume that an external system reports onset times x_k with associated (D)etection precisions γ_k^D and discrete classes c_k (with values for bass, snare and hihat). The precisions represent possible errors in the detection process, not in any high-level sources of note deviations such as changes in tempo or performance timing errors. Once reported, x_k, γ_k^D and c_k are modeled as fixed. Note that a more fully Bayesian system would model them as random variables, allowing prior information from other parts of the music model to modify the onset detector's beliefs. In particular, priors from rival rhythm models

could affect the observations, i.e. rival models' parameters would become dependent on each other. This would greatly increase the complexity of inference and it does not appear to give any practical advantages over independent models, as our task is ultimately to select one winning model. Forwarding messages to rival models without the usual co-parent dependency is performed by the special *multiplexer* nodes in the observation layer.

3.1. Switchboard

We want to infer the positions of notes and beats from the set of observations. If each observed note x_k arrived tagged with a marker saying which note n_j of the model it was an instantiation of, this would be easy, as we could then connect the observed x_k to n_j and perform inference. Hence x_k would be treated as a noisy observation of n_j (as in the systems of Cemgil and Raphael). However we do not have such tagging information, we only have the class of the observation. We are told that it is, say, a bass drum note, but not which of the (many) bass drum notes in the bar. Further, we want to allow the possibility that non-model notes may be played. This may occur for several reasons: (1) extra note played by accident, e.g. the performer's hand slipping; (2) performer is playing a variation of a known model, with deliberate extra notes; (3) performer is playing an unknown model. We must try to perceive the latter as a known model as best we can for now, then perhaps consider the known model's failings and update the model or learn a new model from them offline.

We solve the tagging problem by modeling the belief in the observation locations as a Gaussian Mixture Models (GMMs), with one Gaussian source from each of the M model notes of the same class as the observation. The middle shaded layer of fig. 2 is called the switchboard, and consists mostly of clusters of nodes called switchers. (There are also some extra nodes which are shared across switchers.) One switcher is created per incoming observation k, and includes a GMM node, g, together with a discrete mixture weight node Λ, so that its prior is

$$P(g|\{\mu_m\},\{\gamma_m^P\},\{\Lambda_m\}) \propto \sum_{m=1}^{M} \Lambda_m \exp\left[-\frac{\gamma_m^P}{2}(g-\mu_m)^2\right]$$

where γ_m^P is a (P)erformance precision parameter for the mth parent (see section 3.2). This distribution is not in the conjugate-exponential family, but may be approximated:

$$P(g|\{\mu_m\},\{\gamma_m^P\},\{\Lambda_m\}) \propto \sum_{m=1}^{M} \exp\left[-\frac{\Lambda_m\gamma_m^P}{2}(g-\mu_m)^2\right]$$

This moment-matches the original GMM, and has the useful property that it tends towards it as $\Lambda_m \to \Delta_{mm'}$, that is, as the discrete mixing vector becomes a single peak at m'. This is useful for our task, as we wish to ultimately assign each observation to a single class.

All Λ nodes of each class c share a fixed Dirichlet prior, Dir^c, which have hand-set parameters favouring single-source explanations.

3.1.1. Sibling rivalry

In addition to this GMM formalism, we know something more about the relationship between the observed and model notes: each model note will be instantiated at most once in the data. Even if, say, two snare hits occur very close to each other and to an expected model note location, only one of them can be 'the' instantiation of the note, and the other must be a spurious extra note. The standard GMM is not able to capture this 'sibling rivalry' between its children. GMMs are often used in data-intensive statistics where the task is to infer quantities from large numbers of noisy observations of them. In contrast, here we have at most one observation of each model note. The sibling rivalry problem is not unique to GMMs but is a general Bayes net problem: Bayes nets provide an automatic framework for competing *parents* to inhibit one another through the well-known 'explaining away' effect; but do not allow siblings to compete in any similar way.

To model our knowledge of sibling rivalry, we extend the Bayesian network into a Factor Graph, i.e. a graphical model containing both directed and undirected links. For each note class, we fully connect all the Λ nodes in the switchers of that class. We connect with inhibitory links, so that if one switcher lays claim to being an instance of a model note then it discourages other switchers from being instantiations of it. We extend the variational update for Λ nodes to include this undirected term (rightmost):

$$\log Q(\Lambda) \leftarrow \langle \log P(\Lambda|par(\Lambda))\rangle_{Q(par(\Lambda))}$$

$$+ \sum_{ch(\Lambda)} \langle \log P(ch|\Lambda)\rangle_{Q(ch(\Lambda),cop(\Lambda))}$$

$$+ \sum_{riv(\Lambda)} \langle \log \Phi(riv,\Lambda)\rangle_{Q(riv(\Lambda))}$$

As riv and Λ are discrete distributions parameterized by $Q(\Lambda=\Lambda_m) = q_m^{(\Lambda)}$, $Q(riv=riv_m) = q_m^{(riv)}$ with $\sum_m q_m^{(\Lambda)} = 1$ and $\sum_m q_m^{(riv)} = 1$, we set the additional update terms to

$$\langle \log \Phi(riv,\Lambda)\rangle_{Q(riv)} = \begin{bmatrix} 1-q_1^{(riv)} \\ \dots \\ 1-q_M^{(riv)} \end{bmatrix}$$

where M is the number of components. This is a heuristic whose intent is to encourage pairs of discrete nodes to have large divergences between them (e.g. in the KL-divergence sense). Note that this does *not* assign zero probability to two discrete notes having the same component, because its effect is on the logarithm of the resulting joint, not the joint itself. Instead, it merely discourages such similarity, and we have found it useful enough for our purposes.

Note that this scheme does not force any observations to match up with model notes: it merely discourages multiple observations from explaining the same model note.

3.1.2. Sinks

To allow for the performance of non-model notes, we construct an additional *sink* component for each note type. Sinks may be thought of as null hypothesis explanations for observations. They are modeled by Gaussians having their mean in the center of the bar, and a very wide variance so that the distribution stretches all the way across the bar in one standard deviation. This wide spread has the classic dual effect of (a) catching observations from a wide range, enabling the sink to soak up all observations not well accounted for by other components; (b) assigning a low likelihood to each observation because it is so spread out; hence allowing non-null components to be favoured in their specialist locations.

This may be viewed as a crude approximation to a fuller sink model: there is information contained in non-model notes which we are currently throwing away. In particular, notes do not occur at completely random places in the bar: even if not part of an identifiable model, they still tend to occur on beats or rational subdivisions of beats. In fact many beat identification systems have been constructed in this way. For example Cemgil [1] constructs an intricate prior over subdivisions similar to:

$$P(x) = \frac{1}{Z} \sum_{D=1}^{K} \sum_{N=1}^{2^D} \delta(x; \frac{N}{2^D}) * \exp(-\frac{\gamma}{2} x^2)$$

(The left part of this forms peaks at major subdivisions, whose height increases with importance, as measured by number of factors. The right term convolves (*) with a Gaussian to blur the spikes.) Such priors could in principle be incorporated here as GMMs, with one GMM component per subdivision. The advantage of such a scheme is that tempo and phase may then be inferred in a 'model-less' manner simply from collections of notes and the prior. However this approach would be time-consuming in a real-time system due to the large number of additional components to consider. We also note that 'model-less' is something of a misnomer, as we would effectively be specifying a complete, general pattern based on subdivisions: in practice we can know most common models in advance and exploit this by using our set of particular models instead of this general one. We have found that there is usually enough information in the instances of model notes to infer the beats well, without having to rely on sink notes to improve the accuracy.

There is still one problem with our single wide Gaussian approximation to this intricate structure. As the Cemgil prior is made of peaks, this structure assigns slightly higher priors to sink notes around rational subdivisions than our Gaussian assigns, at the expense of having lower priors than our Gaussian at places away from rational subdivisions. In practice, most (usually all) sink notes *do* occur at rational subdivisions. This means that our approximation will consistently assign lower priors to sink notes than the more accurate structure. To correct for this, we adjust the Dirichlet prior on the Λ nodes to give sinks a slightly higher prior than model notes. Empirically, $u = [0.10, ..., 0.10, 0.12]$ was found to work well, where the 0.12 is the sink component.

As discussed above, we constrain the GMMs with sibling rivalry links to prevent multiple observations mapping to a single model note. For sinks there is no such constraint: we allow multiple observations to be generated by the same sink. To accomplish this, the $\langle \log \Phi(riv, \Lambda) \rangle_{Q(riv)}$ term above is modified so as not to penalize sink-sharing.

3.2. Precisions

The switchboard contains two kinds of (P)erformance precision nodes. First, there is a single fixed precision $\gamma^{P,\emptyset}$ for sinks. This is used as an input to all GMMs, for the precision of their sink component, regardless of their class. It is set so that the standard deviation of the sink is one whole bar. Together with the Dirichlet prior amplification this produces a useful sink model. Second, for each note class c there is a fixed precision $\gamma^{P,c}$ which is shared between all switchers of that class. This performance precision models the player's accuracy in playing notes of class c. For example, the performer may be more accurate on the bass drum than on the hihat. Note that future versions of the system could extend this by allowing each model note to have its own performance precision, in the manner of Raphael's system [3]. This may be useful when the patterns become more complex, and the performer may try harder to play the on-beat notes very accurately at the expense of other more decorative notes. Alternatively, a simplified model might share just one performance precision across all note classes to allow faster inference.

Note that the performance precisions differ from the detection precisions γ^D – the latter being a measure of the detection error rather than the human player's deviation.

3.3. Annealed Variational Message Passing

To reduce the possibility of belief oscillations during message passing, we use variational annealing. In this scheme we do not seek $Q \approx P$ directly, rather we infer a sequence of *heated* distributions $Q^{1/T_i} \approx P^{1/T_i}$. The temperature T begins at a number greater than unity, and is gradually reduced to unity. Further, if an MAP solution is sought rather than a joint approximation, T may be reduced below unity and toward 0, in which case $Q^{1/T}(x) \to \delta(x; \hat{x})$, giving an estimate of the MAP. In practice, this MAP estimate is a near-delta spike at a local maximum of the posterior. As the cooling schedule slows and the initial temperature is raised, the probability of arriving at the global maximum increases. We extend the variational message passing framework to include annealing by giving each node a temperature parameter (for our conjugate exponential models this simply multiplies the parameter vector.)

Fig. 3 shows a difficult three-onset inference problem: the notes fall in such places as to be highly ambiguous under the position and tempo priors. Without annealing this causes the network to oscillate and diverge during inference; annealing cures this problem. Note that the distributions on the nodes begin by becoming very wide as mes-

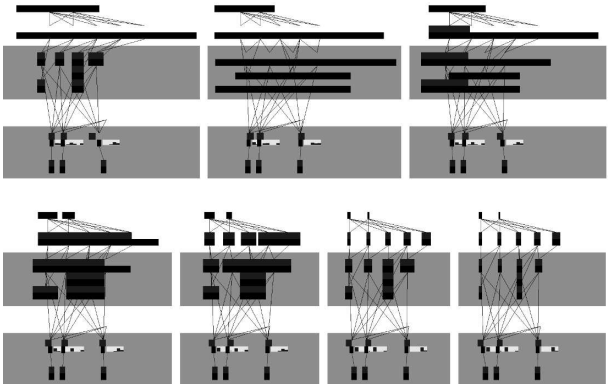

Figure 3. Seven steps of annealed variational inference on an ambiguous three-onset problem. As inference progresses, the heated initial beliefs converge to the correct VB posterior. Without annealing, the network can oscillate and diverge during inference.

sages are received; then gradually reduce as cooling takes place. (See section 5 for a description of the display.)

4. HYPOTHESIS MANAGEMENT

So far we have considered rhythm models of individual bars: test data was created representing a single bar's content of music, and the postulated model had to explain all the data. However in the wild, the situation is more complex: we receive a stream of events over a period of several minutes (the song) and we do not know initially which events belong to which bar and hence to which model.

The ideal Bayesian approach would be to consider all possible models instantiated at all possible points in time-tempo space, together with a prior on which combinations of them are allowable and mixture switches to allow data points to be explained as mixtures of models. The prior would act to allow only sets of rhythm models which are contiguous throughout the whole song, and could further specify Markovian or grammatical priors on the probabilities of particular sequences of models.

This approach is impractical because it requires an infinite number of models, and even quantizing the possible model instances would require a very large number of models. In particular the prior would have to operate on the power set of these models which is intractable. So we must use heuristics to approximate the Bayesian ideal.

The heuristics we use are based on the ideas of priming and pruning presented by [8]. Reis modeled symbolic sequences using a set of *agents,* each agent being a detector for a particular pattern. When an agent *A* successfully completes its pattern (by observing the exact required sequence – Reis' work was non-probabilistic) it competes with completed *rival* agents, and the winner *W* primes all other agents to start looking for their patterns beginning at the point that *W* completed. Rival agents are those which share one or more symbols of the data stream. Losing agents are deleted (*pruned*).

We extend this method to the case of continuous time events (opposed to sequences of symbols). We replace discrete symbol sequence detecting agents with our variational models. Competition between models was handled with a heuristic by Reis but we are able to use the variational lower bound on model likelihood (from section 2) to provide a more rigourous foundation for competition. The principal difficulty is the lack of sharp boundaries between models together with the uncertainty in note arrival times: in the symbolic domain is simply to say 'if the new note time is after the end of agent *A* then add it to the next agent instead of to *A*' but the meaning of this is less clear in our probabilistic domain: there will instead be some probability that the new note lies inside or outside the model. Pragmatically, we simply set a probability threshold and declare the new note to be outside the model if it is exceeded. The pseudocode for our algorithm is thus:

```
prime initial model set
set initial models to incomplete state
for each incoming note n do
  for each incomplete model M do
    if P(n after M < threshold) then
      link n to M
      update temperature T
    else
      set M to completed state
      winner W ← M
      for each rival model R (overlapping M) do
        if R completed then
          if log Q(R|d_R)/|d_R| > log Q(W|d_W)/|d_W| then
            W ← R
            prune R and descendants of R
          end if
        end if
      end for
      prime from W
    end if
  end for
end for
```

where $d_M$is the set of data points attached to model M and $|d_M|$ is the size of this set.

There is an issue regarding model comparison when two rivals *R* and *W* have different numbers of data points. Instead of raw model posteriors, we compare the log posteriors *per data point*, dividing the lower bounds by the number of data. This statistic may be thought of as an approximate log probability *density* per unit time, with data points as a proxy for time. (Note that log posterior densities are typically negative, so dividing them *increases* their value.) The reason for comparing *log* probability per data point is as follows. Suppose we have N models in a sequence, $M_1, M_2, ..., M_N$ which together cover a complete song. The issue is that data points near model boundaries may fall under the model at either side of the boundary. We do not wish to penalize either case in the ideal *global* distribution of models. So we note that our statistic has this property: the score for the whole song is

$$\frac{\sum_{i=1}^{N} \log P(d_{M_i}|M_i)}{\sum_{i=1}^{N} |d_{M_i}|}$$

From the point of view of the global distribution, the denominator is constant so does not penalize any particular assignments of data to models. However recall that we are using a heuristic to approximate the global distribution, namely that we make a hard decision about which models to keep and reject at the end of each bar. And from the viewpoint of these individual bars, the statistic *does* make a difference, namely that of transforming our measure of log likelihood into log likelihood per data point explained, which aids our greedy search through the space of sequences of models as required.

Note that we have included the annealing schedule as part of the node linking cycle. We want the model to start as 'hot' when first created, then to cool as *real* time progresses, so that it reaches the true probability scale as it completes. We use the time of the latest received note as a proxy for elapsed time, and have found that a useful schedule is to cool linearly from $T = 8$ to $T = 1$ during the course of the bar.

4.1. Priming

In the present work we have used a simple Markov transition matrix to specify priors on sequences of models. (e.g. the low probability of a drum'n'bass pattern following a military march pattern.) Note there is an important practical difference between transitions of zero probability and those of small but non-zero probability. Conceptually these are similar, but the former will *not be primed* whereas the latter are primed but are unlikely to be chosen as the final winner. This makes a huge difference to the amount of computation that takes place, but there is the classic tradeoff that in setting unlikely transitions to a hard zero, we rule out the ability to use them to explain uncommon situations, so in the unlikely event that those situations occur, the model will fail. (Primed models are 'known unknowns', as they still contain much uncertainty but are under active consideration'; unprimed models are 'unknown unknowns'.)

Newly primed models are instantiated having a start time with mean equal to the MAP end time of the previous (completed) model – shown by the dotted line in fig. 2 – and inverse tempo mean equal to the MAP of the previous inverse tempo. Parameters specify precisions of these beliefs, which at present are global and hand-set. However it would be conceptually simple to use an EM method as in [3] to learn precisions for each bar and also to learn priors on tempo transitions.

5. EVALUATION AND DISCUSSION

We believe it is more instructive to present a walkthough of a real test run than to give only statistics of its results. The images in figs. 4 and 5 show the result of a complete run on a reasonably challenging data set. The run is quasi-realtime, meaning that simulated time runs at half the speed of the recorded input to allow more message passing cycles. No effort has been made to optimize the interpreted Lisp research code, though the run was performed on a 1.6GHz Pentium with a quasi-time factor of only two which suggests real-time optimization is possible. The data set was recorded by a human drummer using a MIDI kit, and standard deviations of 100ms attached to notes of all types. It consists of 48 bars of rhythms lasting about 2 minutes and including over 250 hits, which is a typical amount of data for beat tracking evaluations, (e.g. [7]). The rhythms begin as simple known patterns and get progressively more complex, and more errors are introduced. A knowledge base of six models was used, including three regular rhythms and three fills. The data begins with a simple four hihat count-in to initialize the priors; the program is initialized with a single count-in model at the correct time and roughly correct tempo, much as a human musician would require. The following description gives a walkthrough of the activity in the first 12 bars only.

Each row of the figures shows four bars. Only the winning models are shown. Each model has the same visual form as fig. 2: at the top left is the s node, to its immediate right is τ^{-1}. Below these are the beats. Time is shown horizontally on each row, and the standard deviation of each node is shown by its width. The τ^{-1} nodes are shown at one beat away from the s nodes. The three shaded areas contain the model notes, the switchboard, and the observations. Within each shaded area are three sublevels for hihat (top), snare (middle) and bass (bottom) – the same format as fig. 1(b). Each switchboard node has a small bar-chart on its right. This shows the value of its switch node where the nth value is the nth model note of the matching type with the rightmost being the sink. All nodes have upper and lower halves showing their prior and posterior respectively – the deviation from the prior is visible only in s and τ^{-1} when inference has converged.

Bars 1 and 2 form a pair of `rock1` and `rock2` patterns. There is a high prior on such a transition. Note that in bar 1, an additional bass drum hit is played on beat 4. This is is not part of the pattern so is assigned to the sink. In bar 4 we see a performance of the highly syncopated `fill-offbeat1` pattern: four snare hits played between the beats instead of on them. This kind of pattern is likely to confuse conventional beat trackers that operate by looking for correlations and phase of onsets and assuming beats match strong onsets. But the system knows this pattern is a fill, albeit with a low occurrence prior, and is able to infer the beat positions correctly.

Bars 5 and 6 see another standard `rock1-rock2` pair. Note the change in s belief in bar 5: the posterior (lower half) s is later than one standard deviation away from the prior (upper half), due to the player loosing time when recovering from the offbeat fill pattern. Bars 7 and 8 contain a further `rock1-rock2` pattern, but in bar 8 the snare hit has been intentionally played half a beat late: it is assigned to the sink (the rightmost bar in its barchart). A possible extension here could be: instead of modeling such

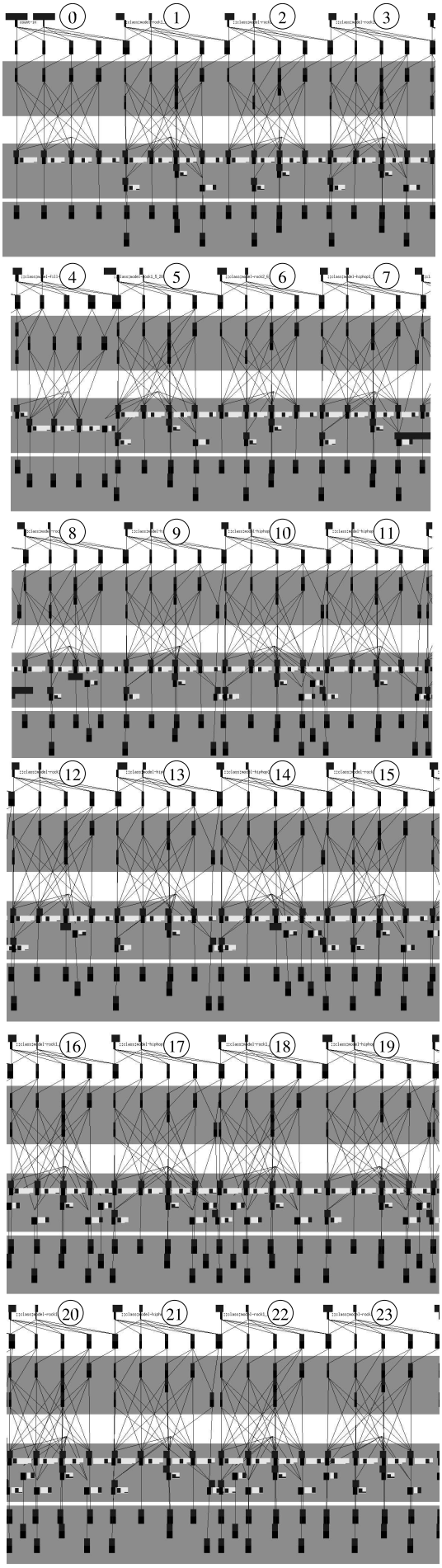

Figure 4. Bars 0-23 of the test run. Bar 0 is the hihat count-in. Bar numbers are shown above the center of each bar.

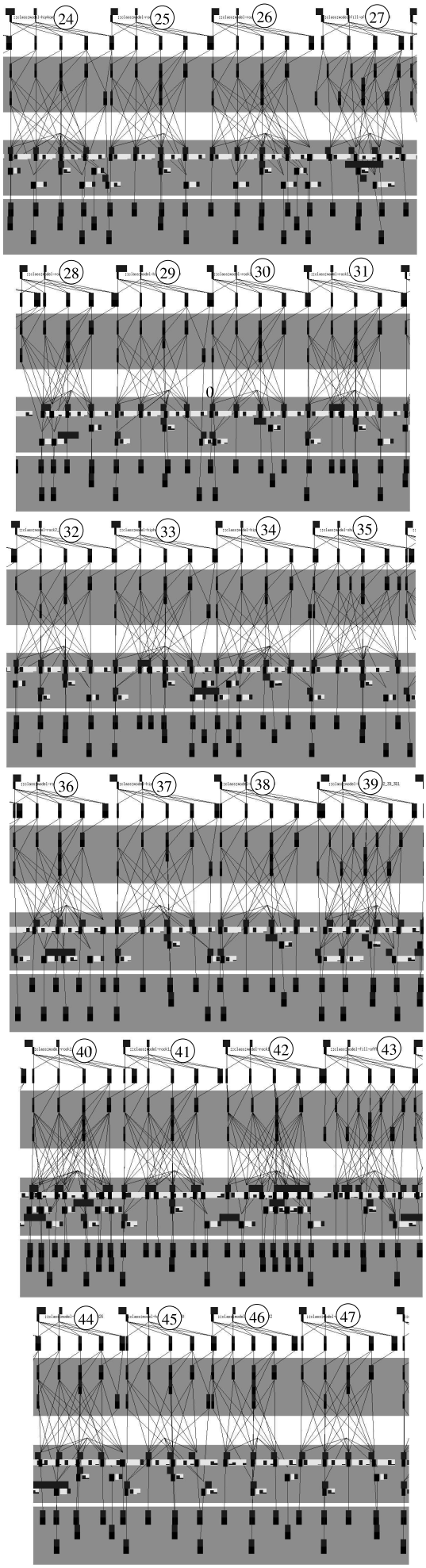

Figure 5. Bars 24-47 of the test run.

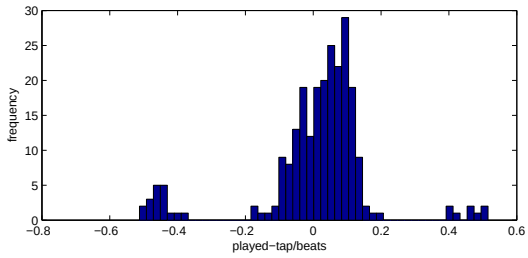

Figure 6. Error of played onsets from hand-tagged beats.

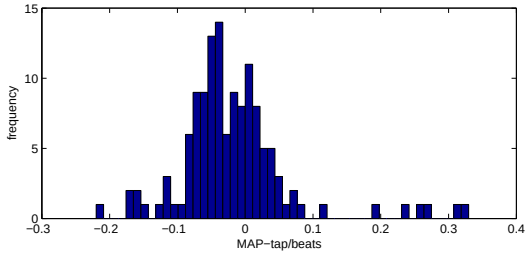

Figure 7. Error of played onsets from auto-tagged beats.

notes from the sink, we might explicitly model them as played intentionally late; a human listener primed to expect `rock2` is likely to hear the note in this way and in fact it is this feeling of lateness that creates the laidback mood of this pattern, sometimes called a 'lounge' beat.

Bars 9-14 switch to a hiphop feel, due to extra hits at beat 4.5. Note again the late snare in bar 12, beat 3.5. It is interesting to consider the 4.5 hit in the same way as in the extension discussed above: currently the system just knows the whole pattern `hiphop1` which contains this 4.5 hit. But humans (especially if unfamiliar with hiphop) may initially hear the 4.5 hit as the first beat of the next bar but played early – in the same way that the 3.5 snare was heard as being from beat 3 but late. In hiphop, the 4.5 hit is quickly followed by a 5.0 hit marking the true start of the next bar. But for half a beat there is a brief feeling that the music has jumped forward a little, before being snapped back. (Perhaps it is this feeling that gives this rhythm its characteristic mood and edgy dancing moves? Again, future work could try to model these ideas.)

The reader may examine the rest of the run, noting the near-losses of tracking around bars 26-28 and 36 and subsequent recovery; and the eventual failure due to complex unknown rhythms at the end.

Fig. 6 shows the deviation of the played onsets from 'ground truth' beats (as labeled by hand, offline), and can be seen as a measure of the accuracy of the playing. The x axis measures fractions of a beat deviation from the nearest labeled beat. Note the clusters around +/- 0.5 beats which are notes played half way between beats. Excluding these notes, the standard deviation (player error) from the beat is 0.07 beats, corresponding to 35ms at 120bpm.

Fig. 7 shows a histogram of errors of the system's MAP beat positions from the ground truth. Only the correctly-tracked bars 1-38 are included. The standard deviation of the errors is 0.12 beats, which corresponds to a musical semi-quaver, or 60ms for a track played at 120bpm. However this large deviation includes the outliers shown in the graph which arise during the near-loss of tracking around bar 27. Excluding these incorrect models gives a deviation of 0.056 beats over 129 data points, equivalent to 28ms at 120bpm. This is a significant (a χ^2 test gives $P(s^2 \leq 0.056^2 | \sigma^2 = 0.07^2, N = 129) = 0.0004$) improvement from the 35ms player latency above (for example if the raw onsets had been used as beat locations.) Typical music software has a latency of around 10ms in comparison. Inspection of the worst errors shows them to occur in bars where the tempo changes within the bar rather than at the barline, this is of course expected as our current model assumes a single tempo within each bar.

While our proof-of-concept implementation of the variational and blackboard ideas described here is not yet at contest standard, we have explained how and why the approach is useful for solving higher-level rhythm and beat perception tasks than those tackled by current beat-trackers. By incorporating psychological ideas of model-based perception, priming and a global workspace, the approach may also lead to clarifications of perceptual music psychology concepts and could even serve as a framework for empirically modeling human rhythm perception, especially for interesting ambiguous percepts.

6. REFERENCES

[1] A. T. Cemgil, H. Kappen, P. Desain, and H. Honing. "On tempo tracking: Tempogram Representation and Kalman filtering" *JNMR*, 28:259-273, 2001.

[2] N. Orio and F. Dechelle. "Score Following Using Spectral Analysis and Hidden Markov Models" *Proceed. ICMC*. 2001.

[3] C. Raphael. "Music Plus One: A System for Flexible and Expressive Musical Accompaniment" *Proceed. ICMC*. 2001.

[4] J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies and M. Sandler. "A tutorial on onset detection in music signals." *IEEE Trans. Speech and Audio Proc.* 13(5):1035-1047, 2005.

[5] J. Winn and C. Bishop. "Variational Message Passing" *J. Machine Learning Res.* 6:661-694, 2005.

[6] M. Goto. "An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds" *JNMR*, 30(2):159-171, 2001.

[7] N. Collins. "DrumTrack: Beat Induction from an Acoustic Drum Kit with Synchronised Scheduling" *Proceed. ICMC*. 2005.

[8] B. Reis. "A multi-agent system for on-line modeling, parsing and prediction of discrete time series data" *Intelligent Image Processing, Data Analysis and Information Retrieval*, IOS, Holland, 1999.