# REINFORCEMENT LEARNING FOR LIVE MUSICAL AGENTS

*Nick Collins*
University of Sussex
N.Collins@sussex.ac.uk

## ABSTRACT

Current research programmes in computer music may draw from developments in agent technology; music may provide an excellent test case for agent research. This paper describes the challenge of building agents for concert performance which allow close and rewarding interaction with human musicians. This is easier said than done; the fantastic abilities of human musicians in fluidity of action and cultural reference makes for a difficult mandate. The problem can be cast as that of building an autonomous agent for the (unforgiving) realtime musical environment. Live music is a challenging domain to model, with high dimensionality of descriptions and fast learning, responses and effective anticipation required.

A novel symbolic interactive music system called *Improvagent* is presented as a framework for the testing of reinforcement learning over dynamic state-action case libraries, in a context of MIDI piano improvisation. Reinforcement signals are investigated based on the quality of musical prediction, and on the degree of influence in interaction. The former is found to be less effective than baseline methods of assumed stationarity and of simple nearest neighbour case selection. The latter holds more promise; an agent may be able to assess the value of an action in response to an observed state with respect to the potential for stability, or the promotion of change in future states, enabling controlled musical interaction.

## 1. INTRODUCTION

Interactive music systems [25] are software and hardware systems founded on AI techniques which are designed for music-making, most typically in live concert performance combining machine and human musicians. Contemporary work in this field includes investigations into both machine listening (realtime audio analysis) and robotics; an inspiring project in this regard is Ajay Kapur's *MahaDeviBot*, a thirteen armed Indian percussionist which can synchronise to sensor input from a human sitarist [19].

Recent years have also seen a number of such projects intersecting with the agent community, from Belinda Thom's *Band-out-of-the-Box* [30] and Wulfhorst and colleagues' *Virtual Musical MultiAgent System* [32], to the *Musical Acts - Musical Agents* architecture [23], *OMax* [1] and Arne Eigenfeldt's *Drum Circle* [10]. Indeed, agent technology has a potential to influence the general field of computer music, as discussed by Dahlstedt and McBurney [6], a composer and agent researcher who have collaborated on generative music software. Perhaps the earliest explicit live musical agent work is that of Peter Beyls, whose 1988 description of *Oscar* (Oscillator Artist) [3] characterised the system as an autonomous agent. [1]

A goal of this research is the realisation of autonomous agents for interactive music, which can at a minimum operate independently of composer intervention during performance, though they may not be so independent of the composer's programming. Behavioural autonomy in a concert and rehearsal situation (self sufficiency of action) is sought whenever the agent is switched on, but constitutive autonomy (continuity of existence at the scale of everyday life) is not expected [12]. To quickly align with musical demands, techniques for fast adaptation to musical situations must be explored.

This paper will proceed by more closely examining work to date on musical agents. Because machine learning is identified as a shortcoming of much existing work, we will investigate the combination of music and reinforcement learning techniques adopted by the agent community. A new MIDI based system will be described, intended as a testbed for experiments in online adaptation.

## 2. AGENT ARCHITECTURES FOR MUSIC

A natural viewpoint in applying agent metaphors to music is to place the agents at the level of individual musicians such that each concert participant is a single autonomous agent. This will be the primary level at which agency is discussed in this paper, but individual artificial musicians as multiagent systems have also been considered. Minsky's society of mind metaphor has been applied by Robert Rowe particularly in his *Cypher* project; the Meta-Cypher includes multiple listener and player agents as well as a Meta-Listener [26, pp. 310-15]. In the setting of artificial life, Jonathan Impett has demonstrated the complex emergent musical interaction possible with swarm intelligence [18]. But multiagent architectures within individual artificial musicians have usually been notional, for instance, as simple active hypotheses in computational beat tracking [13].

Where systems have sought to explore flexible interaction with human musicians, style specific cultural conventions and innate human musical behaviours (such as synchronisation abilities) have provided severe challenges for

[1] Though it fails to qualify under more restrictive definitions [4].

researchers. In an analysis of the state of the art interactive music systems were characterised with respect to taxonomies of agent systems [4]. Building an autonomous agent for live performance, with as much musical autonomy and flexibility as a human performer, was found to be restricted in current generation systems by insufficient appropriate musical training and pro-activeness. Arguably, current generation musical agents cannot meet strong conditions of agency as outlined by Wooldridge and Jennings [31]; this may be arguable in turn for so many systems, situated on a continuum from object to human agent.

Indeed, much research [2] has been guilty of anthropomorphism, as pointed out by Wolfhurst et al [32], who themselves proceed to make claims for 'leadership' and 'happiness' amongst their musical agents, and describe an agent component which is essentially a MIDI device manager. In the context of rhythmic agents in a 'drum circle' [10] many parameters are described in terms of social music making, the agents being 'confident' or 'mischievous'. Beyl's Oscar is somewhat anthropomorphised in his description of its 'personal opinion' [3], and there is a two dimensional state for the system on axes of interest (from bored to aroused) and stimulation (from under to over stimulation) based on the pitch content of working memory. Such descriptors are useful indicators of the designer's intention, and music is an often ambiguous and always social art that naturally gives rise to such circumstances of intentional stance; but semantic musical labels from the researcher will not necessarily lead to enhanced artificial intelligence.

An agent perspective may be a beneficial stance to adopt in music AI. A clear exemplar of the intersection of music modelling and agent technology is the MAMA architecture described by Murray-Rust and co-authors [23], which proposes a musical agent communication protocol of 'Musical Acts' analogous to 'Speech Acts'. Multiple communication channels are theorised, covering both musical and extra-musical gesture. This structure is applied in a description of interactions for the classic minimalist work *In C*, to allow generative performances by musical agents of the work within the bounds of its rule scheme. This system is not a learning system, however, and we now turn to the issues of machine learning in music.

## 3. MUSICAL AGENTS WHICH LEARN

The extensive practice regimes of human musicians, whether the obsessive study of popular musicians [14] or conservatoire training regimes of thousands of hours [7], point to a critical role for machine learning in machine musicianship. Though David Cope claims there are 'few published examples of computational learning in music' [5, p. 181] in his most recent book, there are many examples of machine learning systems which he fails to cite, most notably in the contemporary research area of music information retrieval. The research interest of MIR tends to be directed most strongly towards the semantic web, with the general audio consumer in mind; our research is directed most at the general practising musician.

Whilst restricted to the limited domain of 4 bar call and response against an imposed metronome, Belinda Thom's work on 'improvisational music companionship' [30] applies machine learning technology in specialising an interaction to an individual musician. By post-session analysis of material collected during rehearsal, *BoB* can adapt to a given player , collating (by unsupervised clustering of data) a set of playing modes meant to represent different styles of performance. Thom's investigation of machine learning techniques that might be applicable to the 'sparse' material offered in musical dialogue is noteworthy, and she even contends that 'perhaps sparsity can be used to model musical creativity'. This is probably underestimating the amount of practice a musician has engaged in through their life [3] , though it is a pragmatic approach to train an interactive system, and the only situation encountered in immediate short-term interactive applications.

Perhaps the most advanced work to date in the emulation of human musicianship is that of Hamanaka and collaborators [15]. They train *virtual players* from the performance of a trio of human guitarists, learning independent models for reaction (within ensemble playing behaviour), phrase (a database of individual musical materials) and groove (timing preference). A particular offline algorithm accompanies each of these, respectively, radial basis function networks mapping between subjectively assigned impression and intention spaces, Voronoi diagrams for phrase segmentation, and a HMM for timing parameters. The authors effectively tackle the problem of sparse data for learning, but are somewhat locked to the source data for their experiments: that initially gleaned from the human guitarists. Aspects of their conception have influenced this paper and the system described below, and whilst they treat imitation more than generalisation and learning, the set-up and vision they describe is extremely appealing. For bootstrapping especially, their solution may provide excellent techniques, but it remains untested in more general applications.

There is a question of the role of such systems; is it realistic to expect an interactive music system to learn from the ground up during concerts? At the very least, musical knowledge is built into such 'general' systems in their internal representations and rules. Human musicians have usually been bootstrapped through all sorts of musical educational experiences before they step out at Carnegie Hall. Most learning systems will be trained offline, that is, on pre-existing corpora, with appropriate ground truth for supervised learning. Musicianship provides an interesting challenge in combining some amount of bootstrapping, unsupervised learning, and online tuition and concert experience.

Some researchers have approached the demanding task of within concert learning by variable length Markov models, which can develop quickly in realtime interaction, the

[2] I am not exempt from such criticism, having used rather anthropomorphic parameter names for the Free Improvisation Simulation [4]

[3] 7000 hours or so for a professional standard violinist by age 18 [7]

Continuator being a famous example [24]; an alternative is the efficient Factor Oracle algorithm [1]. Such rote learning with generalisation from pattern matching fits well to the demands of immediate response, but has been criticised by François Pachet himself as 'reflexive' rather than 'flexible' music making; these algorithms have more limited higher level musical facility.

## 4. REINFORCEMENT LEARNING AND MUSIC

The machine learning technique of reinforcement learning [28] has many links to the needs of realtime agent systems, due to its inherent approach to complex learning environments. Only a few authors have previously considered the application of these techniques to computer music.

Franklin and Manfredi [11] study actor-critic reinforcement learning, using a nonlinear recurrent neural network for note generation in a context of jazz improvisation. The reinforcement signal is derived from eight hand-written rules specifying a basic jazz music theory. They also provide some limited results on three tasks concerning simplified musical sequences.

In the closing stages of their paper on the OMax system, Assayag and colleagues [1] discuss an experimental application of discrete state-action reinforcement learning to weight links in the Factor Oracle algorithm. When this takes place from live interaction, positive reinforcement corresponds to extra attention to certain musical materials (the improviser having no way however to show any dislike of the material generated by the algorithm). A 'self-listening' mode is also investigated where the system feeds back its generated material to itself, this time with negative reinforcement so as to increase variety in productions.

Reinforcement learning requires a musically salient notion of reward, and as in general research into musical agents, it is essential to derive some measure of the quality of musical actions for particular situations. A number of candidates have been proposed; such functions have been variously defined as fitness functions for genetic algorithms, utility measures, or reinforcement signals. Murray-Rust et al. [23] list three feedback measures on agent performance:

1. Matching internal goals (for example, the imposition of a rule set as per [11])
2. The appreciation of fellow participants (and further, of a general audience and critical authorities)
3. Memetic success; the take-up of the agent's musical ideas by others (within a concert and a culture)

The second of these is difficult to measure during a concert, for the entry of values at a computer interface may disrupt the flow of performance for a performer; an independent observer logging state may not have access to a musician's real state of mind (post concert analysis is even more troublesome, as Thom notes concerning the difficulty of retrospective inquiry into improvisation [30]). Camera tracking of facial expression and attention, as well as physiological monitors such as galvanic skin response sensors, might provide reward measurements mirroring instantaneous engagement, but have not been explored.

Variations on these ideas might look specifically at musical phenomena, for instance, comparing individuals to an ensemble by the metrical synchronisation of a performer within an ensemble, the reconciliation of pitch materials to the overall harmonic consensus or timbral appropriateness. In applying the third criteria above, measures of the information content and musical similarity of statements and answers might seek to explore how 'inspiring' given productions were. But it should be noted that musical goals and ambiguity can vary widely in different stylistic settings; for example, the short-term alliances and dissolutions of free improvisation as well as the rapid turnover of material can make measurement difficult.

Having seen some of the potential pitfalls, a pragmatic approach might be to use predictive success as the reward signal itself. Musical interaction is inherently based on anticipation [17]; human musicians cannot react within milliseconds, but can adapt within half a second to two seconds by reconfiguring their expectancy field. Given the current state of the musical environment, the machine musician can posit a prediction of the next state (on which its own scheduled productions will draw). After the next state measurement, the system is in a position to assess how successful this prediction was; further, the timescale of this feedback fits the online and local nature of the task.

The second reinforcement signal investigated in this paper is similar to the third listed above, though at a shorter timescale and with less high level a description; it concerns the perceived effect on the position in state space of the action taken. How much can an agent influence the current status quo through the choice of action?

Before we move onto a specific system, however, it must be acknowledged that reinforcement learning itself has been criticised as too slow a process for online adaptation, particularly in situations with large parameter spaces.

Investigating variants of reinforcement learning, Dinerstein and colleagues [8, 9] provide algorithms tailored to the needs of fast adaptation for autonomous game characters. In the treatment below, the Sarsa($\lambda$) algorithm for discrete state-action spaces is the primary technique used, but the dynamism of recording case libraries and comparison with nearest neighbour methods is motivated by their ideas, as a pragmatic tackling of the high dimensional interactive music domain.

## 5. IMPROVAGENT

This section of the paper will describe a system which has provided a prototype for reinforcement learning experiments. The working title of the system is *Improvagent*, for 'improvisation agent', or more hopefully, 'improving agent', describing a musical agent which seeks to improve itself through constant learning. Whilst many of this authors' previous interactive music systems have been audio analysis based [4], a symbolic MIDI system was designed for this research project, so as to more directly tackle the essential questions of agency and learning. Since the MIDI specification is most suited to piano, Improvagent combines a human pianist with a virtual player. The agent is equipped for both realtime learning during interaction with a human pianist, and non-realtime learning from MIDI files (especially for boot strapping and for non-interactive testing), as an online reinforcement learning process in both cases.

A paradigmatic and challenging case for interactive music systems is that of improvisation. It is no longer necessary even in musicological circles to defend improvisation as a practice; for instance, Derek Bailey [2] and George Lewis [20] have already done an admirable job here, and the ubiquity of improvisation in human musical culture is now readily acknowledged. However, grandly claiming improvisation as the domain of a musical agent is to gloss over the wide variety of improvisational practice. Through the assumptions of the MIDI protocol and 88 keyed 12TET piano, to Western music theory notions of key and metre that will underlie the working version of Improvagent, the scope of the system is circumscribed. Restrictions on the current abilities of the system will be readily acknowledged, and Improvagent stands as much as anything as a test case for the adaptation mechanisms.

The timescale of operation of this system is designed to fit human processing constraints of perceptual present and reaction time. Half second frames form the basic block of processing, though with reference to a wider history of the last two seconds (four frames at this rate) for such measures as key and metre. In the current working system, computational beat tracking is side stepped; instead a tempo of 120bpm is imposed (so that a beat exactly matches the frame period). This is a practical step taken to avoid unnecessary complexity in this study, and the extension to the case of beat tracking can be envisaged.

Over the course of a frame, all MIDI note events are stored. Three viewpoints on the current frame are maintained in line with realtime data acquisition; a list of new onsets, a list of note off events, and a list of active MIDI notes at the end of each frame. Even with the assumption of an imposed metronome, there are issues of chord onset asynchrony and quantisation. Experiments in measuring chord onset asynchrony observed inter chord note interval of around 5 msec in tight playing, whereas looser playing allowed up to 30 msec spread (more for truly sloppy playing!). To accommodate this, events within 30 msec of the end of a frame are deferred to the next frame, as anticipations of the beat. Any further chord tones close to these (within 30 msec of another note in the closing region) are themselves deferred in turn. Such deferral, alongside the tracking of MIDI off messages and active notes, is important to accurate measurement of harmonic content.

## 5.1. Feature extraction and proximity measure

The foundation of Improvagent is the treatment of successive frames as successive observed environmental states.

| Parameter | Implementation | Values |
|---|---|---|
| num onsets | classes: 0, 1-2, 3-4, 5-6, 7+ | 0-4 |
| key type | major, minor or 'neutral' | 0-2 |
| transpose | transposition of original pitch materials | 0-11 |
| keydist | conflict of key calculated over the window versus the local key over the frame | 0.0 to 1.0 |
| register type | harpsichord, full piano, just bass, just top | 0-3 |
| max pitch | highest appearing note | 21-108 |
| min pitch | lowest appearing note | 21-108 |
| groove | straight or swung | 0 or 1 |
| lead or lag | mean expressive deviation ahead or behind beat | float |
| active notes | active notes during the frame | list |
| new onsets | notes starting during the frame | list |
| max vel | highest velocity value in frame | 0.0 to 1.0 |
| action | assigned to following state or tested action | pointer |
| value | rating of this state-action pair | 0.0 to 1.0 |
| framestart | original recorded time of state | time |
| p c profile | pitch class profile over current window | 12 floats |
| q profile | allocation of onsets to quantisation positions weighted by velocity | 4 floats |
| expressive deviation | sum of absolute deviations from quantised rhythm over window | float |
| density | notes active, new onsets per frame | 2 integers |
| register | tessitura, median pitch | 2 integers |

**Table 1**. Table of state data and features

Cases are typically derived from pairs of a state and its successor, assuming that the human provides a model of useful actions [8]. Table 1 details the program data associated with a state. The table is broken down into four sections; the first corresponds to parameters used to partition the set of states so as to reduce computational load in search, as detailed below. The second concerns features which are stored and may be used in the generation of response material, but which essentially come along for the ride. The third specifies parameters for the case. The final division concerns those features used directly in the proximity measure for matching states. For the proximity measure, relevant features are normalised to an equivalent 0.0 to 1.0 parameter range.

Key extraction [26] used a comparison to each possible transposition of templates for major, minor and various 'neutral' chord types (whole tone and chromatic aggregate scales). The highest scoring match to the input pitch class profile gave the key type and transposition. The pitch class profile itself took account of proportional durations and amplitudes of notes within frames in weighting individual note contribution.

Beat extraction is itself trivial in the case of an imposed metronome, but evaluating expressive timing compares the observed onset positions within the window to a quantisation template for two groove patterns: straight and swung semiquavers. The template with the smallest overall total time deviations in seconds was selected as best fitting the current evidence, and the groove and deviation features then follow directly; the lead/lag parameter is a measure of the mean error of quantised location to original timing, and gives some indication of playing ahead or behind of the beat.

A proximity measure between states is defined by a Euclidean distance over a feature vector of the final features in the table (with user specifiable weightings of the dimensions; equal weighting for experiments herein). For the case of comparing the pitch class profiles of states s and t (each vectors over the 12 chromatic pitch classes) the distance contribution from this dimension is the square of the Manhattan metric between profile vectors:

$$\text{dist}(s,t) = \min(\sum_{i=0}^{11} |\text{pcp}_s(i) - \text{pcp}_t(i)|, 40) * 0.025 \quad (1)$$

where pcp is the unnormalised pitch class profile formed over the window, and the minimum operation keeps things constrained for cases of very high note densities as compared to normal operation.

The set of features described here are hardly exhaustive, and one might immediately investigate further contour and pitch interval representations (perhaps a set of measures between onsets separated by n notes, for n = 1 to numonsets-1), which maintain more of a sense of strict temporal ordering. Indeed, measures of melodic similarity of actual note sequences might replace Euclidean distance between feature vectors in the proximity measure [16], following more complicated case-based reasoning systems [22, p. 240]. A lot depends on the level of abstraction sought for the musical system in its engagement.

Whilst this prototype treats the case of discrete MIDI events, aspects of this investigation may be pertinent to audio based interactive music systems, where features may be sampled at a higher rate across frames; again, the selection of state data may include features for state proximity measurement, variables for learning and generation, and some discrete factors to assist the breaking down of the size of the problem domain.

## 5.2. Case storage

One approach to controlling the explosion of dimensions is to split the total database into a number of parallel case libraries, keyed by important musical parameters. The current system uses three key categories (major, minor and neutral) and five onset count categories (0, 1-2, 3-4, 5-6, 7+), for a total of 15 parallel databases.

Search for matches is therefore constrained from the outset by these categories accurately reflecting the situation, but does not have to take place over every previously observed case. Furthermore, cases are never stored if both the state and following state are themselves empty of events; this avoids the accumulation of cases when the human is not playing.

The working hard limit on the number of cases per database is 1000, corresponding to 8 minutes 20 seconds of material, giving an overall memory for the system of around two hours, assuming even distribution amongst the 15 parallel case libraries. There is a rule for replacement, invoked if the number of existing cases in the target database is already at the limit. The state with maximum score is determined with respect to a sum of three evenly weighted factors of age, inverse distance to the incoming state, and inverse value. This procedure (with the addition of the value) follows Dinerstein and Egbert [8].

Counts of the accumulation of cases for a typical live training run over 1359 recorded frames (over 11 minutes of playing, not including unrecorded double zero state-actions) follow: [ 30, 78, 360, 175, 51, 14, 29, 92, 80, 60, 28, 47, 88, 102, 125 ]. The largest peaks correspond here to major key material with 3-6 new note onsets per frame; the take-up is non-uniform, but still utilises each library. Certain categories might be granted greater storage, though the distribution can vary widely as well between runs depending on what material is performed (different results are obtained for contemporary music improvisation and romantic and later MIDI files!). Further, the hard limit is of the cost of search as well as overall memory constraints.

## 5.3. Reinforcement learning algorithms

Fast adaptation demands the fast assimilation of musical ideas by the system; state-action pairs are dynamically added with each successive frame. Because the possible cases are not fixed in advance, a k-nearest neighbours algorithm ensures a dynamic (lazy learning) approach, following Dinerstein and Egbert [8]. The operational algorithms of Improvagent use a k-NN linear search at their heart, a search which cannot be improved by a *k*D-tree since the database is itself being continually updated. Value guides policy by determining the selection of the highest value case amongst the k nearest neighbours of the query state;[4] updating value is accomplished by reinforcement learning steps.

Variants of Sarsa($\lambda$) were utilised over the acquired cases (state-action pairs) to investigate musically motivated reinforcement learning with respect to two potential reinforcement signals. The first of these, 'prediction', operates by immediate feedback on observing a successor state; the second, 'consequence' is driven by the next but one state. In the central loop of both algorithms, MIDI note data in the current frame is parsed, the enclosing window updated, and musical features are extracted as per section 5.1. At this point the two algorithms differ in their detail due to the different delays before reward calculation.

### *5.3.1. Prediction*

1. Create new state s' from features extracted in frame
2. Match s' using k-nearest neighbours (with respect to a feature based proximity function) within an existing state database; store k predictions of next state
3. Top rated prediction is new action a'
4. Compare prediction a from last frame s to observation s' (Sarsa reinforcement of $s_1$)

[4] Additional options would be to form interpolants of the neighbours by mean or median [8, 9], with distance and value weighting.

5. Compare other predictions $a_i$ from last frame s to observation s' (immediate reward update of $s_i$)
6. Update last frame state s with action s' and add to database of state-action cases, removing old state-action pairs if necessary
7. Store new state s' by assigning s = s'
8. Generate machine output based on prediction a'

This algorithm is a pragmatic combination of a number of ideas; the use of k-nearest neighbours and the update of the value of multiple states follows [8], feature matching is related in audio research to concatenative synthesis [27] and the Sarsa algorithm is described in [28].

In Improvagent, Sarsa($\lambda$) updates value for a case $v(s_i, a)$:

$$v(s_i, a) = v(s_i, a) + \alpha((1.0 - d(a, s')) + \gamma v(s'_1, a') - v(s_i, a)) \quad (2)$$

where $d(s, t)$ is the proximity measure between feature vectors. The reward thus favours a close proximity of prediction and new observed state. Eligibility traces are maintained in a list holding the last ten touched cases for further back updates; various values of the parameters were investigated, defaults being $\epsilon = 0.1$, $\lambda = 0.9$, $\alpha = 0.5$, $\gamma = 0.5$. Experiments were also carried out with 'side predictions' by retaining the top K neighbours as additional predictions; a recursive step is not justified here for any update, so $\gamma$ and $\lambda$ were zero.

Figure 1 plots an observed state succession $s \rightarrow s' \rightarrow s''$ and three nearest neighbours $s_1$ to $s_3$ of s in two dimensions; such implicit paths in the feature space continue on for the sarsa estimate (with $s'1 \rightarrow a'$) in particular. In this predictive situation, $a1$ as the primary prediction is less successful than $a2$ from a side prediction; $v2$ will be boosted, whilst $v1$ is estimated downwards except for the additional sarsa bonus for $v(s', a')$.

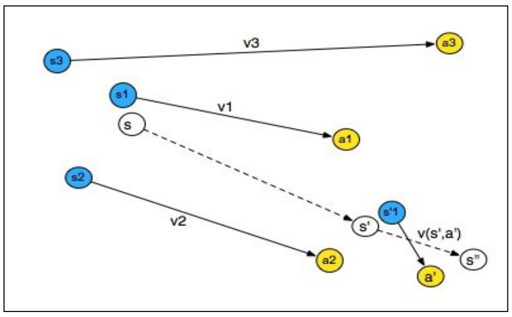

**Figure 1**. Nearest neighbours and predictions for observed $s$ and $s'$

Convergence properties of both algorithms were approached empirically, because of the k-NN and dynamic case aquisition potentially interfering with known convergence properties of discrete on-policy Sarsa($\lambda$). Runs of the system were carried out in which each case maintained a record of when and how much it had been updated over the course of interaction; counts, mean distance from state to current observed state and mean change in value were tracked over state-action cases.

Despite any early bias with a small number of cases spread out in feature space, wild oscillation of values was not observed. However, the musical setting is the primary consideration, and a number of factors count against general convergence, not least of which is the wilful exploration of the human musician; just because a particular state led to another earlier in the improvisation does not guarantee exact repetitions! We would expect, however, a certain amount of repetition appropriate to the style.

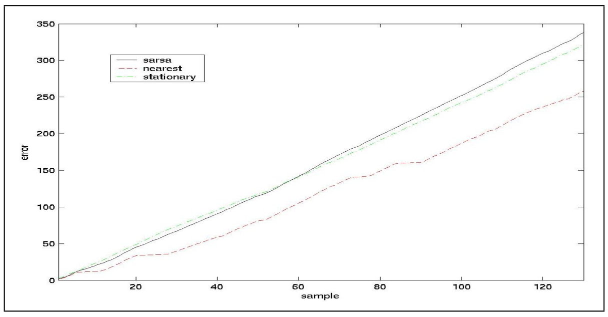

**Figure 2**. Error of prediction algorithm against baseline algorithms

The primary goal is an accurate prediction of the next state; if the state is rarely encountered, so be it, and the rare state may be deallocated in favour of newer or more valued states. This is absolutely fine and natural; we don't keep so much in our memories that wasn't successful or useful! If there is a worry over maintaining some representative rarer states, a periodic assessment of divergence and coverage in the state space might be carried out, and certain states ear marked for preservation. Yet, the potential state space is so much larger than any encountered in practice (due as always to music's innate combinatoriality) that a sparse set of representatives are potentially all that can be maintained.

We have discussed the algorithm here at length to demonstrate how a practical musical system might go about utilising reinforcement learning. Unfortunately, one flaw remains: there are better predictors of immediate successor state than the reinforcement learning algorithm itself. The baseline comparators were first, using the current observed state as the prediction (assumption of stationarity) and second, using the nearest neighbour in the case library without following the highest value. Figure 2 gives a counter example over training by seven MIDI files, with an accumulated case library of 1302 frames (10 minutes 51 seconds). Similar results were seen for live improvisation, with the stationary assumption often edging out the simple nearest neighbour in sessions. This demonstration undermines the prediction algorithm in its current form and motivates a new twist.

### 5.3.2. *Consequence*

In an attempt to leverage the technical mechanisms, but provide a more effective measure for interactive improvisation, additional delay between act and reward was intro-

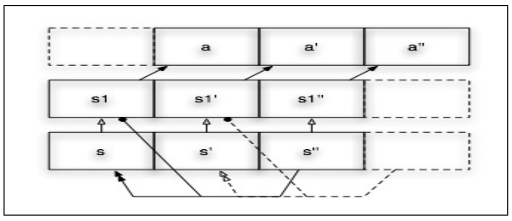

**Figure 3**. State-action cases and propagation of information for consequence assessment

duced. The reinforcement signal now measures the repercussions or consequences of a musical action in response to an observed state; that is, it considers a single moment of interaction between human player and machine agent. The algorithm is:

1. Create new state s" from features extracted in frame
2. Match s" to k-nearest neighbours (with respect to a feature based proximity function) within an existing case library
3. Top value case state $s_1$" provides new action a"
4. Compare frame s to observation s" and update v($s_1$,a) (Sarsa($\lambda$) reinforcement with eligibility traces)
5. Add (s,s') to the database of state-action cases, removing old state-action pairs if necessary
6. Advance states assigning s' = s", s=s'
7. Generate machine output for upcoming frame s"' based on prediction a"

Three successive states are involved; given s, an action a is selected which runs against s', the moment of interaction; s" is then measured against s. The algorithm thus compares the actual change against a baseline of stationarity; the underlying assumption is that the players are paying attention to each other, and action a can be assessed as promoting either change or stasis if following the context state s. Intermediate state s' is assumed to be underway before the player has time to react to the machine action a; the full consequences are only really apparent by state s". Figure 3 expresses this in diagram form; the middle line of states s1, s1', s1" are those winning states from the case library (highest value except for $\epsilon$-greedy random selection). The comparison of s" to s is shown at the bottom of the figure, and the rising line back to state s1' indicates the update to the value. The sarsa($\lambda$) update to value for the case $v(s_1, a)$ is now:

$$v(s_1,a) = v(s_1,a) + \alpha(d(s,s'') + \gamma v(s_1'', a'') - v(s_1,a)) \tag{3}$$

so that the reward is a measure of state change; stationary behaviour corresponds to zero reward. Eligibility traces are maintained as before, but only one action can be evaluated at a time and there are no side predictions or other simulations.

This algorithm provides a mechanism for measuring the machine's influence in interaction, which can be learnt during interaction; indeed, it is intimately bound to interactive settings itself, and makes no sense in training over fixed and irresponsive MIDI Files. In testing, the set of features and proximity measure chosen, as well as the interactions observed, were a factor in how well overall the system anticipated the consequences of an action in a given situation. Further evaluation is needed, especially over alternative timescales of consequence, as well as algorithm variants.

### 5.4. Material generation

Because the set of features and recorded data for a frame provides both abstract and exact description of the events of that frame, there is a choice of response. For many interactive music systems, following the gist of density and pitch material data might be appropriate. In prototyping, Improvagent was set to playback the stored notes precisely, or to select from a subset of active notes. Material was thus scheduled in advance of the new beat.

An optional facility was added to play back a chain of states in a row (by following successive action pointers, which replay a sequence in time). For prediction this depended on the value of the primary prediction, as a confidence relative to the second best prediction and proximity of states [5] .

Conventional Sarsa($\lambda$) updates cannot take place during longer sequence playback; playback is always curtailed if the cumulative error of match between sequence location and current observation exceeds a constant, if a predetermined number of steps are exceeded, or if a state has no valid action (which may occur if cases have been replaced in the libraries). Such longer term material generation raises questions of the negotiation of intention between human player and machine.

## 6. DISCUSSION

The choice of musical representation has a critical effect upon the system. Improvagent calculates a number of pertinent musical features, but has no wider ability in segmentation (to separate out voices and phrases), longer time windows and hyper-metrical information, stylistic analysis and the automatic recognition of and specialisation for certain genres, or indeed, physiological data on human pianism. Yet even though the musical model has weaknesses, the recognition system relies on similar musical materials mapping to similar states; triplet figures mistakenly taken under the swung groove will happen relatively consistently with respect to such categorisation. If harmonic materials are octotonic, and might be classified as neutral or minor, the most important thing is that the mapping is dependable, such that further examples of such materials will end up in roughly the same place; continuity of state-action underlies the stable learning of value.

Alternative reinforcement learning mechanisms (i.e., function approximation), higher level actions and multilayered learning [8] have not been broached in this pa-

[5] It may be possible to explore self-simulation by combining prediction of a player's responses and cumulative assessment of consequence

per but are definite targets of future study. In the musical setting, one might ask when the human player would like support, and when stimulation/opposition? What are the dynamics of systems of multiple virtual players, each of which can learn during the encounter? How might improvisation structures such as John Zorn's 1984 game piece Cobra be instantiated? Such high level negotiation has been explored by the MAMA system [23] but is not straight forward to formalise at the level of task and goal selection as per [8]. It is expected though that there may be dividends in the application of state-action-reward methods in modelling such exchanges; measurements of mutual influence and the consequence of an action within a group setting are future areas of investigation.

## 7. CONCLUSIONS

Live music is a domain which can provide an imposing test of agent technology. This paper considered a system for online value attribution via reinforcement learning to dynamically acquired state-action cases, founded on musical feature spaces. Concertising demands fast adaptation, and whilst boot-strapping and careful representational choices can simulate practice, runtime value assignment and learning assists specialisation to and accommodation of human musicianship by machines. It is interesting to speculate on the needs of musical agents which might train with their human owners over multiple practice sessions and performances.

Whilst the issue of other modalities of communication than sound have not been tackled in this paper, further directions for systems emulating musicians are found in embodied virtual agents, which take advantage of senses other than audio (such as video feeds) to gauge an interlocuter's intentions, and are typically instantiated as 3-D rendered characterisations evidencing behaviours [29, 21]. The next stageis to consider higher level musical and extra-musical intention; undoubtedly, live musical agents which learn are a stimulating endeavour providing much to consider for the computer music community.

## 8. REFERENCES

[1] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov. OMax brothers: a dynamic topology of agents for improvization learning. In *AMCMM '06: Proceedings of the 1st ACM workshop on audio and music computing multimedia*, pages 125–132, 2006.

[2] D. Bailey. *Improvisation: Its Nature and Practise in Music*. Moorland publishing Co Ltd, Ashbourne, Derbyshire, England, 1980.

[3] P. Beyls. Introducing Oscar. In *Proc. Int. Computer Music Conference*, 1988.

[4] N. Collins. *Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems*. PhD thesis, University of Cambridge, 2006.

[5] D. Cope. *Computer Models of Musical Creativity*. MIT Press, Cambridge, MA, 2005.

[6] P. Dahlstedt and P. McBurney. Musical agents: Toward computer-aided music composition using autonomous software agents. *Leonardo*, 39(5):469–70, 2006.

[7] I. Deliège and J. Sloboda. *Musical Beginnings: Origins and Development of Musical Competence*. Oxford University Press,, New York, 1996.

[8] J. Dinerstein and P. K. Egbert. Fast multi-level adaptation for interactive autonomous characters. *ACM Trans. Graph.*, 24(2):262–288, 2005.

[9] J. Dinerstein, P. K. Egbert, and D. Ventura. Learning policies for embodied virtual agents through demonstration. In *IJCAI*, pages 1257–1262, 2007.

[10] A. Eigenfeldt. The creation of evolutionary rhythms within a multi-agent networked drum ensemble. In *Proc. Int. Computer Music Conference*, Copenhagen, Denmark, 2007.

[11] J. A. Franklin and V. U. Manfredi. Nonlinear credit assignment for musical sequences. In *Second international workshop on Intelligent systems design and application*, pages 245–250, 2002.

[12] T. Froese, N. Virgo, and E. Izquierdo. Autonomy: a review and a reappraisal. In *Proceedings of the 9th European Conference on Artificial Life*, 2007.

[13] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–71, 2001.

[14] L. Green. *How Popular Musicians Learn*. Ashgate, Burlington, VT, 2002.

[15] M. Hamanaka, M. Goto, H. Asoh, and N. Otsu. A learning-based jam session system that imitates a player's personality model. In *IJCAI: International Joint Conference on Artificial Intelligence*, pages 51–58, 2003.

[16] W. B. Hewitt and E. Selfridge-Field, editors. *Melodic Similarity: Concepts, Procedures and Applications (Computing in Musicology II)*. MIT Press, Camb, MA, 1998.

[17] D. Huron. *Sweet Anticipation*. The MIT Press, Cambridge, MA, 2006.

[18] J. Impett. *Computational Models for Interactive Composition/Performance Systems*. PhD thesis, University of Cambridge, 2001.

[19] A. Kapur, E. Singer, M. S. Benning, G. Tzanetakis, and Trimpin. Integrating hyperinstruments, musical robots & machine musicianship for North Indian classical music. In *Proc. NIME*, pages 238–241, 2007.

[20] G. Lewis. Too many notes: Computers, complexity and culture in *Voyager*. *Leonardo Music Journal*, 10:33–9, 2000.

[21] M. Mancini, R. Bresin, and C. Pelachaud. An expressive virtual agent head driven by music performance. *IEEE Transactions on Audio, Speech and Language Processing*, in press, 2007.

[22] T. Mitchell. *Machine Learning*. McGraw-Hill, Singapore, 1997.

[23] D. Murray-Rust, A. Smaill, and M. Edwards. MAMA: An architecture for interactive musical agents. In *ECAI: European Conference on Artificial Intelligence*, pages 36–40, 2006.

[24] F. Pachet. The Continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–41, 2003.

[25] R. Rowe. *Interactive Music Systems*. MIT Press, Cambs, MA, 1993.

[26] R. Rowe. *Machine Musicianship*. MIT Press, Cambs, MA, 2001.

[27] D. Schwarz. *Data-driven Concatenative Sound Synthesis*. PhD thesis, Université Paris 6, 2004.

[28] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[29] R. Taylor, D. Torres, and P. Boulanger. Using music to interact with a virtual character. In *Proc. NIME*, pages 220–223, 2005.

[30] B. Thom. BoB: an interactive improvisational music companion. In *AGENTS '00: Proceedings of the fourth international conference on autonomous agents*, pages 309–316, 2000.

[31] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.

[32] R. D. Wulfhorst, L. Nakayama, and R. M. Vicari. A multiagent approach for musical interactive systems. In *AAMAS '03: Proceedings of the second international joint conference on autonomous agents and multiagent systems*, pages 584–591, 2003.