Nonlinear Credit Assignment for Musical Sequences

Judy A. Franklin **Victoria U. Manfredi**
Computer Science Department
Smith College
Northampton, MA 01063
jfranklin@cs.smith.edu vmanfred@cs.smith.edu

Abstract

Reinforcement learning is an exciting possibility for generating music because of its ability to learn without explicit examples and to produce more than one response in a given state. We use reinforcement learning in the second phase of a jazz improvisor that learns to interactively play jazz with a human. The reinforcement signal is based on rules for improvisation. Because of time delays between note played and subsequent reinforcement, a critic adjusts the reinforcement signal. We describe this system and then examine the ability of a temporal difference critic to predict reinforcement for three different sequential musical phenomena. A nonlinear network with a linear TD output unit and context traces on input is able to successfully predict reinforcement values for these sequences and shows promise for use in musical reinforcement learning tasks.

1 Foundations

Jazz improvisation is the creation of a jazz melody in real time. Charlie Parker, Sonny Rollins et al. were founders of bebop and post bop jazz [11] where drummers, bassists, and pianists keep the beat and maintain harmonic structure. Other players improvise over this structure and take turns improvising for 4 bars at a time. This is called trading fours. We have been researching neural network-based improvisors. Artificial neural networks have been used in computer music [14, 16] and e.g. Todd [15] used a recurrent network to learn to reproduce short classical songs and then produce new ones. Todd's work is the basis for the first phase of our two-phase learning system. First, a recurrent network is trained with back-propagation to play three jazz melodies by Sonny Rollins [1]. This network is a modified version of the network Todd developed and it has traded fours with a human player in real time. We will describe it briefly in Section 2. In phase 2 an actor-critic reinforcement learning configuration enables the machine to use the first network but to improve its improvisation ability by trying actions and then receiving reinforcements for them, according to a set of rules (Section 4). We examine these rules and then consider a subset of sequential prediction problems covered by these rules. We show results of prediction of reinforcement for these sequences when using a TD-based nonlinear critic with function approximation accomplished via a nonlinear neural network. This work is described in Sections 4 and 5. We include an appendix (Section 6) on a basic description of jazz improvisation.

2 Phase 1

In Phase 1, supervised learning is used to train a recurrent network to reproduce the three Sonny Rollins melodies. The network is a Jordon net [7] with linear output units and nonlinear hidden units (logistic function). Chords provide the harmonic structure of a song, and the chord over which current notes lie is input to the network (Figure 1). 24 of the 26 outputs are notes (2 chromatic octaves), the 25th is a rest, and the 26th indicates a new note. The output with the highest value above a threshold is the next note, including the rest output. The new note output indicates if this is a new note, or if it is the same note being held for another time step (16^{th} note resolution). More details of this phase are given in [4, 5]. The network learns to produce each of the three songs as well as all three songs in sequence. Success is measured by noting dramatic decreases in the squared error as well as comparing the music score of the ANN version of the song with the original. And of course listening to the song provides aural feedback as well. In trading fours

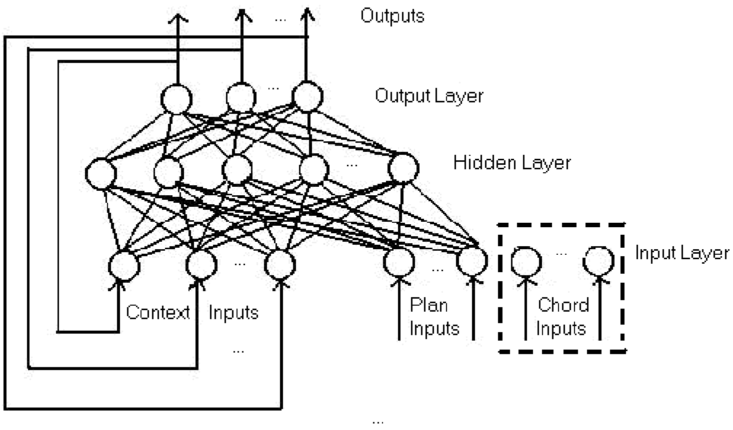

Figure 1: Jordan recurrent net with addition of chord input

with the trained network, human note events are brought in via the MIDI interface [9]. Four bars of human notes are recorded and then given, one note event at a time, to the context inputs (replacing the recurrent inputs). The plan inputs are all 1. The three songs are similar in harmonic structure and the chord inputs follow one of them. The machine generates its four bars and they are played in real time. Then the human plays again, etc. An accompaniment (drums, bass, and piano), produced by Band-in-a-Box software (PG Music), keeps the beat and provides chords for the human to hear. The interaction has been analyzed by following the resulting musical score (again, see [4, 5]). The machine plays certain notes in places where they appear in the original songs. There are notes that appear quite frequently in the Rollins songs and so appear often in the machine's music. The rhythm (produced indirectly by choosing rests instead of notes) follows the human's by changing slowly when the human's previous 4 bars are changing slowly, and changing quickly when the human's does so.

3 Phase 2

In Phase 2, the network is expanded and trained by reinforcement learning to improvise according to the rules of Section 4 and using its Phase 1 knowledge of the Sonny Rollins songs. Using actor-critic reinforcement learning ([2, 12, 17]), the actor chooses the next note to play. The critic receives a reinforcement signal r from the critique made by the rules. Figure 2 shows the Phase 2 network. The figure shows a configuration in which the critic is "piggy-backed" onto the nonlinear network, and uses the features learned from the action network. The same inputs plus 26 human inputs used during training and trading fours brings the total to 68. The weights obtained in phase 1 initialize this network. The reinforcement learning algorithm used is called SSR [3, 5]. It is a real-valued output algorithm derived in part from the prior work of [6, 17]. The piggy-back configuration also alleviates the computational burden for real time operation. In a typical example using the phase 1 network prior to phase 2 improvement, the average reinforcement value -.37 (on a scale from -1 to 1). After Phase 2, the average reinforcement value is .28 after 30-100 off-line presentations of the human solo of 1800 note events. In the resulting machine improvisations, the note durations are shortened, reflecting the rules to prevent settling onto one note. The machine plays chord tones, retains notes used heavily in Rollins' melodies, and due to its recurrence, produces a recurring motif with small variations, an artifact of a "good" jazz solo. The phase 2 network has been used to interact with a human in real time while still learning. It keeps its recurrence since the human has a separate set of inputs.

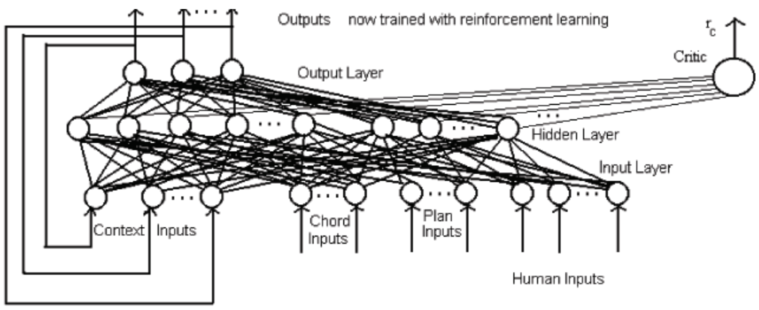

Figure 2: Phase 2 network with critic “piggy-backing” on hidden layer.

4 Nonlinear TD network

The rules used to critique the actions and produce the reinforcement signal are described here and in the appendix within the framework of a very basic description of jazz theory. Using reinforcement values of very bad (-1.0), bad (-0.5), a little bad (-0.25), ok (0.25), good (0.5), and very good (1.0), the rule set is:
1) Any note in the scale associated with the chord is ok (except as noted in rule 3).
2) On a dominant seventh, hip notes 9, flat9, #9, #11, 13 or flat13 are very good. One hip note 2 times in a row is a little bad. 2 hip notes more than 2 times in a row is a little bad.
3) If the chord is a dominant seventh chord, a natural 4th note is bad.
4) If the chord is a dominant seventh chord, a natural 7th is very bad.
5) A rest is good unless it is held for more than 2 16th notes and then it is very bad.
6) Any note played longer than 1 beat (4 16th notes) is very bad.
7) If two consecutive notes match the human’s, that is good.
8) Any other note is bad.
The machine’s improvisations are encouraging yet indicate a need to study critics’ ability to predict reinforcement for isolated musical phenomena. Even without understanding the numerical music notatation, we can see that the critic is required to count, in a fairly sophisticated manner. In a tabular state representation, a single weight value is associated with each state and a separate TD predictor is assigned to each state. The problems that arise are the combinatorial explosion of state entries with increasing complexity of state, and no generalization across states. As an alternative, a function approximation tact can be taken in which some function of the states is used as the prediction. In the simplest case a linear TD algorithm makes a linear approximation of the value of each state. It is our conclusion after re-examining these rules and experimenting with TD(λ) that a nonlinear TD with a neural network front end can predict reinforcement for the counting task. In addition to counting, we decided to explore two other predictive musical tasks for which TD could be useful, in which we use TD as the output unit of a nonlinear network and within a non-Markov problem. The two phenomena are 1) the use of passing tones and 2) forward motion. In the passing tones task, a note is played that must be resolved in one or two time steps by a note that ”works over” the current chord (c_i). The task for TD is to predict a high reward when it sees the first unresolved note. In the forward motion task, the network is given two c_i inputs. The first, c_1, corresponds to the current chord and the second, c_2 corresponds to the next chord to be played. TD must predict a high reward if a note is played at the end of a sequence over a chord c_1 that it does not work with, if the note it *does* work with is present on the second chord input, c_2. Jazz improvisors will do exactly this. While a chord is still being played by the rhythm section, an improvisor will play a note that sounds ”bad” at first. But when the next chord is played, it is resolved and sounds even better than if a ”normal” note had been played over the first chord. In fact both of these phenomena involve the rewarding of the introduction of musical tension, followed closely in time by its release.

5 Results and Conclusions

We experimented with several approaches in using the TD algorithm in a gradient-descent based configuration with variations in how the state is represented and where eligibility traces are used (decaying moving average filters). We

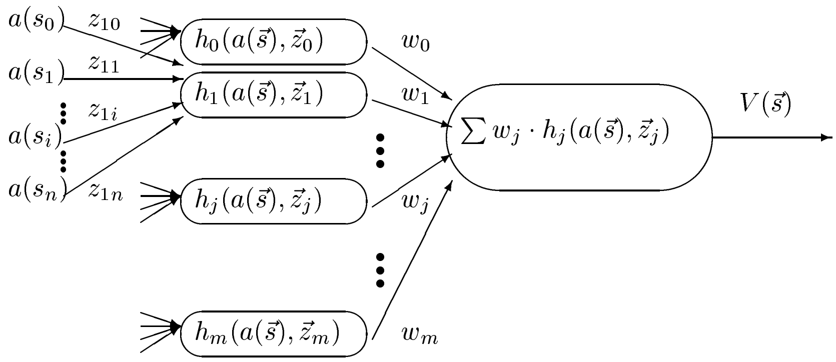

Figure 3: Nonlinear TD network showing moving average on inputs

found the most successful to be a linear output unit that makes the prediction V(s) the linear weighted sum of the outputs of the hidden layer units. TD is cast as the output unit of a gradient descent-based network [13, 8] minimizing, for state vector s

$$E = 1/2(V^{\pi}(s) - V(s))^2 \tag{1}$$

where the TD prediction (output) is

$$V(s) = \Sigma_{j=0}^{m} w_j h_j(s, z_j). \tag{2}$$

Here, $h_j(s, z_j)$ is the output of the j^{th} hidden unit and z_j is the vector of weights that connect the state vector s to the hidden unit. These units are just the oft used vector dot product passed through the logistic function. The w_j are the weights connecting the hidden units to the TD output unit and they are updated using the TD algorithm shown here in vector form as

$$\Delta w = \alpha \delta e(h(s, z)). \tag{3}$$

$\delta = r + \gamma V(s^{'}) - V(s)$ is the TD error that is an approximation of $(V^{\pi}(s) - V(s))$ and the eligibility that is the λ distribution of reinforcements over time, also in vector form, is

$$e(h(s, z)) = \lambda e(h(s, z)) + \nabla_w V(s) = \lambda e(h(s, z)) + h(s, z). \tag{4}$$

We also use a λ eligibility trace on the weights (vector z_j) connecting the state to each hidden unit, leading to the weight update

$$\Delta z_j = \alpha \delta e_{h_j} \tag{5}$$

$$e_{h_j} = \lambda e_{h_j} + \nabla_{z_j} V(s) = \lambda e_{h_j} + w_j h_j(s, z_j)(1 - h_j(s, z_j))a(s). \tag{6}$$

Because the logistic function is used, $h^{'}(s, z_j) = h_j(s, z_j)(1 - h_j(s, z_j)$. The last term in equation 6, $a(s)$ is used rather than s because the sum used in calculating each h_j is the dot product of the decaying averages $a(s)$ of the state inputs and the weight vector z_j where $a(s) = \beta a(s) + (1 - \beta)s$. The TD eligilities are mechanisms for distributing the reinforcement signal over the weights, over time. And the moving average trace, $a(s)$ makes available a decaying history over the past several time steps of the state inputs for feedforward prediction in these non-Markov tasks.

Nonlinear Counting Experiments

In this experiment, 20 sequences of five notes are used. 10 of the sequences consist only of note two (22222) and 10 sequences consist of two occurrences of note two randomly distributed within the sequence and three other random notes (e.g. 23254). A set of 5 note indices are also given to the network, indicating where the current note is in the sequence (e.g. 00001 for note 1, 00100 for note 3). A reinforcement of 1 is given on the last note in the sequence if the sequence is 22222; otherwise a reinforcement of zero is given. Weights are initialized between -.33 and .33. 60 hidden nodes are used and $\alpha = .02$, $\beta = .6$,$\lambda = .6$, and $\gamma = .95$. The following shows 4 typical sequences after training and shows TD predicts high reinforcement as long as all values are 2:

2	V = 0.92 r = 0	1	V = 0.08 r = 0	1	V = 0.08 r = 0	2	V = 0.92 r = 0	1	V = 0.08 r = 0
2	V = 0.90 r = 0	2	V = 0.43 r = 0	3	V = -0.29 r = 0	2	V = 0.90 r = 0	5	V = -0.13 r = 0
2	V = 0.89 r = 0	6	V = -0.37 r = 0	5	V = -0.33 r = 0	5	V = 0.11 r = 0	6	V = -0.46 r = 0
2	V = 0.89 r = 0	2	V = -0.21 r = 0	2	V = -0.18 r = 0	4	V = 0.09 r = 0	7	V = -0.40 r = 0
2	V = 0.87 r = 1	4	V = 0.07 r = 0	2	V = 0.23 r = 0	8	V = 0.12 r = 0	4	V = -0.56 r = 0

Nonlinear Ordered Sequence Experiment
In this experiment 20 sequences of 5 notes are presented. There are 60 hidden nodes and chords are still not used. If the sequence of 012 occurs, the reward of 1 is given at the last note in the sequence. Otherwise a reward of 0 is given. The 012 sequence is presented one third of the time the time and otherwise values are presented in random order (e.g. 251). Weights are initialized between -.05 and .05, $\alpha = .02$, $\beta = .6$, $\lambda = .6$, and $\gamma = .95$. Typical results after training are below. TD undeniably distinguishes 012 from 345. On 342, it "favors" the 2 in the last position:

0	V = 1.54 r = 0	2	V = 1.91 r = 0	3	V = 1.20 r = 0	3	V = 1.20 r = 0
1	V = 1.81 r = 0	1	V = 1.62 r = 0	4	V = 1.03 r = 0	4	V = 1.03 r = 0
2	V = 2.14 r = 1	0	V = 1.83 r = 0	2	V = 1.59 r = 0	5	V = 1.00 r = 0

The result is somewhat subtle when comparing the similar sequences 210 to the favored 012. TD's ability to distinguish them appears in its prediction reinforcement for the value 2, which is significantly higher when following 01.
Forward Motion - Switching Chords
In this experiment the network is presented with 4 consecutive 3 note sequences. The note 0 works well over the chord c_1=10001010010 (chords are represented as 12 inputs, 4 of which have value 1 and correspond to the 4 notes of the chord). However, 0 does not work well over chord c_2=101000100100. The number of presentations is 60,000 (a larger number of rounds produced oscillating behavior), there are 20 hidden nodes, 22 different sequences are presented, and 3 different chords are presented (and the notes in the chords are mututally exclusive). Weights are initialized to values between -.05 and .05 and $\alpha = .02$, $\beta = .6$, $\lambda = .6$, and $\gamma = .95$. Below is a table showing 6 consecutive sequences. For each sequence, 2 chords are given, e.g. chord c_1 for the first sequence and c_2 for the next chord. The network is given both chords as inputs and the table of results below shows these 2 chords for each sequence. 0 appears as the last note of sequence two (10 is the first note of the third sequence). We can see 0 in this position in columns 2, 5, and 6. TD predicts high reinforcement for 0 in this position regardless of the next chord. However, sequences over the chord c_2 following the sequence with 0 (see column 3) receive a higher predicted reward than others. So TD has determined that 0 in position 3 is good and that chord c_2 is good after a 0 but has not managed to put them together quite as we desire yet.

	c2,c0		c0,c2		c2,c0		c0,c1		c1,c0		c0,c1
9	V = 1.39 r = 0	11	V = 1.34 r = 0	10	V = 1.50 r = 0	11	V = 1.45 r = 0	1	V = 1.48 r = 0	6	V = 1.33 r = 0
2	V = 1.37 r = 0	1	V = 1.33 r = 0	2	V = 1.48 r = 0	6	V = 1.40 r = 0	11	V = 1.48 r = 0	6	V = 1.33 r = 0
5	V = 1.36 r = 0	0	V = 1.70 r = 0	9	V = 1.47 r = 0	4	V = 1.37 r = 0	0	V = 1.88 r = 0	0	V = 1.73 r = 0
11	V = 1.33 r = 0	10	V = 1.51 r = 1	11	V = 1.44 r = 0	1	V = 1.33 r = 0	6	V = 1.62 r = 0	6	V = 1.45 r = 0

Our goal here was to study how to use the TD reinforcement prediction algorithm with a nonlinear network that backpropagates the TD error. Furthermore, we pushed this combination by presenting problems that are non-Markov prediction problems based on musical phenomena. We would like to pursue the last experiment, context-switching, to ijmprove the results, but overall see this use of TD as a promising one for the generalization of reinforcement learning in domains that require function approximation and that contain non-Markov tasks. Experimentation continues for these problems as well as experimentation on using the SARSA algorithm to learn to generate the sequences.

6 Appendix: A Basic Tutorial and Rules for Jazz Improvisation

The harmonic structure of a jazz song is a series of chords (groups of notes played simultaneously). Chords differ by the number of steps or half-steps between the notes as they appear in the chromatic scale. Several important chords are the major triad, minor triad, and the diminished triad, all being 3 notes separated by a set number of half-steps called thirds. A third added to one of these triads forms a seventh chord: the major seventh chord (F-A-C-E is F major seventh or Fmaj7), the minor seventh chord (e.g. Fm7 is F-Ab-C-Eb (Fm7)), and the dominant seventh chord (F7 is F-A-C-Eb), respectively. These chords are used heavily in jazz harmony.

A scale, a subset of the chromatic scale, is characterized by note intervals. E.g., the F major scale is F-G-A-Bb-C-D-E-F. The notes in a scale are degrees; E is the seventh degree of F major. Roman numerals represent scale degrees and their seventh chords. Upper case implies major or dominant seventh and lower case implies minor seventh [11]. The major seventh chord starting at the first note of a scale is the I (one) chord. G is the second degree of F major, and G-Bb-D-F is Gm7, the ii chord, with respect to F. The ii-V-I progression is prevalent in jazz [11], and for F it is Gm7-C7-Fmaj7. The minor ii-V-i progression is obtained in a similar manner. Most jazz compositions are either the 12 bar blues or sectional forms (e.g. ABAB, ABAC, or AABA) [10]. The 3 Rollins songs are 12 bar blues. "Blue 7" has a simple blues form. In "Solid" and "Tenor Madness", Rollins adds bebop variations to the blues form [1]. ii-V-I and VI-II-V-I progressions are added and G7+9 substitutes for the VI and F7+9 for the V; the II-V in the last bar provides the turnaround to the I of the first bar to foster smooth repetition of the form. The result is at left and in Roman numeral notation at right:

Bb7	Bb7	Bb7	Bb7
Eb7	Eb7	Bb7	G7+9
Cm7	F7	Bb7 G7+9	C7 F7+9

I	I	I	I
IV	IV	I	VI
ii	V	I VI	II V

One way a novice improvisor can play is to associate one "standard" scale with each chord and choose notes from that scale when the chord is presented in the musical score, whence Rule 1. Next, the 4th degree of the scale is often avoided on a major or dominant seventh chord (Rule 3). The major 7th is avoided even more on a dominant seventh chord (Rule 4). Rule 2 contains many notes that can be added. Seventh chords can be extended by adding major or minor thirds, e.g. Fmaj9, Fmaj11, Fmaj13, Gm9, Gm11, and Gm13. Any extension can be raised or lowered by a half step [11] to obtain, e.g. Fmaj7#11, C7#9, C7b9, C7#11. The C7 in Gm7-C7-Fmaj7 may be replaced by a C7#11, a C7+ chord, or a C7b9b5 or C7alt chord [11]. The scales for C7+ and C7#11 make available the flat 5, and flat 6 (flat 13) for improvising. The C7b9b5 and C7alt (C7+9) chords and their scales make available the flat9, raised 9, flat5 and raised 5 [1]. These substitutions provide the notes of Rule 2.

References

[1] J. Aebersold. *You can play Sonny Rollins. A New Approach to Jazz Improvisation,8*. Jamey Aebersold, New Albany, IN,1976.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive element that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846, 1983.

[3] H. Benbrahim . J. Franklin. Biped walking using reinforcement learning. *Robotics and Autonomous Systems*, 22:283–302, 1997.

[4] J. A. Franklin. Multi-phase learning for jazz improvisation and interaction. *Proceedings of the Eighth Biennial Connecticut College Symposium on Arts and Technology*, 2000.

[5] J. A. Franklin. Learning and improvisation. *Neural Information Processing Systems 14, edited by T.G. Dietterich, S. Becker, & Z. Ghahramani*, 14, 2001.

[6] V. Gullapalli, J. Franklin, and H. Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, 1994.

[7] M. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1986.

[8] V. U. Manfredi. *Nonlinear Algorithms for Reinforcement Learning*. Sr. Honors Thesis, Smith College, Northampton, MA, 2002.

[9] P. Messick. *Maximum MIDI*. Manning Publications, Greenwich, CT, 1988.

[10] S. Reeves. *Creative Jazz Improvisation. 2nd Ed.* Prentice Hall, Upper Saddle River NJ, 1995.

[11] M. A. Sabatella. *Whole Approach to Jazz Improvisation*. A.D.G. Productions, Lawndale CA, 1996.

[12] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[13] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[14] N. G. Griffith and P. M. Todd. *Musical Networks: Parallel Distributed Perception and Performance*. MIT Press, Cambridge MA, 1999.

[15] P. M. Todd. A connectionist approach to algorithmic composition. In P. M. Todd and E. D. Loy, editors, *Music and Connectionism*. MIT Press, Cambridge MA, 1991.

[16] P. M. Todd and E. D. Loy. *Music and Connectionism*. MIT Press, Cambridge, MA, 1991.

[17] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.