ALGORITHMS FOR MUSIC COMPOSITION BY NEURAL NETS: IMPROVED CBR PARADIGMS

J.P. Lewis

Digital Sound Laboratory
New York Institute of Technology
Old Westbury, NY 11568

ABSTRACT

Neural networks have proven suitable for problems characterized by "fuzzy" structure that is difficult to describe by rules or algorithms. The facility that neural net approaches have demonstrated in these problems is potentially relevant to computer arts applications including music composition which require the description or generation of patterns having a desired but fuzzy structure.

Current proposals for music composition by neural nets are surveyed, and several computationally feasible variants of the previously intractable *Creation by Refinement* (CBR) paradigm are introduced. One such algorithm, termed *attentional* CBR, is illustrated with a simple problem in music composition.

1. INTRODUCTION

Music (and art in general) is often characterized by a "fuzzy" structure or balance between order and disorder [1] that has proven difficult to describe algorithmically. Many algorithmic methods for the quantitative description of music can be characterized as either statistical (e.g. Markov process approaches [3,4]), or rule-based approaches including generative grammars [6]. The limitations of these methods are complementary: whereas it is expensive to capture global regularity using statistical methods, rule-based methods are poorly suited to describe ambiguity and irregularity, and typically must include provisions to weaken the strength or applicability of the rules [6].

Neural nets or 'connectionist' approaches have proven suitable for problems having perceptually limited or fuzzy structure, and thus appear promising for the machine composition of music and other "artificial creativity" problems.

Unfortunately, direct neural net approaches to most musical problems are not practical on current serial computers. To indicate one benchmark, NETtalk [11] is a well known problem which required overnight runs on a superminicomputer. This problem used a backpropagation net [10] with approximately 200 inputs and 10,000 weights. Using an enumerative representation with each note being represented by 12 inputs to enumerate pitch and perhaps four additional inputs to represent duration, a computationally equivalent problem with 200 inputs would represent only about 13 notes! Thus, researchers hoping to employ neural nets in music must find ways to limit the applicability of the net to a small part of the overall task (or a vague overview), or await the availability of suitable parallel machines.

The paper will survey existing proposals for music composition by neural net, and will propose several computationally tractable variants of the attractive but expensive CBR algorithm. Current proposals utilize the popular backpropagation gradient descent learning algorithm [10] in various ways. This algorithm has been described extensively elsewhere; for the purpose of our description, backpropagation can learn an arbitrary mapping from m inputs to n outputs by example, and, if properly trained, can generalize, producing reasonable outputs from novel inputs.

This ability to generalize suggests applications such as music composition. The problem is to identify the right learning task (function) for the net–some training approaches do not yield learnable functions, or functions on which generalization is musically relevant.

2. CREATION BY REFINEMENT

Creation by Refinement (CBR) [8] is a neural net paradigm developed specifically for artificial creativity problems such as the machine composition of music. CBR consists of a learning phase, in which a supervised gradient descent learning algorithm learns to be a 'music critic' (preferentially judging musical examples according to various criteria), followed by a creation phase, in which a haphazard creation is refined by a gradient descent search until it is judged to satisfy some of the trained criteria. Since CBR relies on an unspecified gradient descent learning algorithm, and any such algorithm can be employed (including non-connectionist algorithms), it is presented as a paradigm rather than a specific algorithm.

In the learning phase of CBR, a number of musical (and non-musical) patterns are presented as input to the net, and the trainer's critique of the corresponding patterns is presented as the desired output of the net. The differences between the net output and the critique are fed back into the net as a training error E, and the net weights W are adjusted by gradient descent in the direction $-\partial E/\partial W$.

The critique must be numerically expressed but may be many-dimensional. A simple example of a training set would be various pitch sequences which are considered musical or non-musical by the trainer, together with the trainer's corresponding categorization, expressed as a 1 or 0. A more complex example might include additional dimensions to describe the style, historical period, or other characteristics of the sequences. Including additional parameters is somewhat risky, since there must be some real relationship between the sequences and how they are categorized in order for the net to have a learnable problem. For example, an age parameter will not be learnable if the training patterns do not contain discoverable features which are correlated with their age.

Following training, the inverse of the learned criteria is probabilistically explored to generate novel patterns: the net input is set to a random vector, and the net output vector is compared to a critique vector representing a desirable creation. The input vector I is then refined by a second gradient descent in the direction $-\partial E/\partial I$, minimizing the difference between the desired and obtained output vectors, and so refining the input to satisfy the desired criteria.

The CBR algorithm is described in more detail in [8]. This algorithm is conceptually satisfying in that the creation process is not represented by a sequence of rigid rules, and in that the criteria learned in training will be fuzzy if the presented patterns require this. As mentioned previously however, CBR is too expensive to handle non-trivial musical problems on serial machines.

3. SEQUENTIAL NETWORK

Several authors [12,2] have proposed using backpropagation to memorize a number of musical sequences, and then use some scheme to interpolate these sequences. In the approach of Todd [12], a sequential or Jordan net is used to associate monophonic melodies with numbers ("plans"). The net can then generate a new melody which combines intervals from the memorized sequences.

The Jordan net [5] is a backpropagation configuration designed to produce a sequence of outputs at regular intervals. To accomplish this, some of the outputs are autoregressively filtered and routed back to the inputs to provide an exponentially decaying memory of the past output of the sequence. One such context input represents the complete sequence of the corresponding output.

This compressed memory format allows the Jordan net to memorize and reproduce fairly long sequences. It has the corresponding drawback that certain structures are not efficiently described [9], and infinite loops result when the net fails to learn these structures. In particular, structures where similar antecedents lead to dissimilar consequents are problematic for an approach based on the recent past. These structures are very common in music, e.g. an ABAC period form, and may be quite long, e.g. an AAAAAAB... ending of an ostinato passage.

The Jordan net is not an obvious choice for music composition. In addition to the consideration mentioned above, the restriction to sequential composition is artificial, particularly when (as in [12]) the model is proposed as a model for human composition (we are certainly not restricted from thinking about how the end will affect the middle, or from composing an ending first). The sequential approach would appear to be much more suitable as a model for a real-time accompaniment process.

The most serious drawback of the Jordan net for compositional purposes is that it is a *deterministic* scheme. While the Jordan net resembles a learned Markov representation, deterministic **(context→output)** transitions are learned, rather than random transitions selected from a specified probability distribution. Todd [12] demonstrated interpolating between existing 'plans' to generate new melodies. This has the effect of selecting among the note transitions of the memorized melodies.[1] For the right plan value, most transitions will be taken in nearly equal amounts from the two adjacent (in plan space) melodies, as demonstrated in [12].

Todd suggests producing melodies which are not simply mixes of memorized sequences, by providing enough examples that the (fixed size) net begins to generalize rather than memorize. Specific **(context→output)** transitions such as **(GGG→Eb)** can be generalized to more general contexts by a suitable training set, but the output will be the mean of the presented outputs rather than a probability distribution. That is, a transition such as **(Frippish things→A#)** will not generate novel music.

4. CBR VARIANTS

4.1. Attentional CBR

In this variation on CBR, a large problem is partitioned into manageable subproblems using an attention mechanism. The context necessary to tie the subproblems together is provided by context inputs, which during creation are clamped (fixed) to the values of the surrounding and previously constructed pattern. As an example, to produce elaborations on a short phrase, the training set inputs would consist of sample phrases together with corresponding embellished phrases (using a suitable null-note representation to allow different phrase lengths), and the training input would (as usual) consist of some critique of the character of the embellishment. In the creation phase, the embellished inputs would be set to random values, but the context inputs would be clamped to the phrase.

The attention mechanism may be provided by a simple algorithm; various compositional approaches are possible depending the selected mechanism. In *sequential* CBR, prior output (in time) provides the context, and the attention is on the next note or phrase. Like [12], this is a restricted compositional approach, but one which is suitable for real-time applications. In *recursive* CBR, different levels of representation are progressively filled in, with the desired level (and recursion scheme, e.g., ABC→AxByC) indicated in the criterion vector. In *interactive* CBR, a human operator selects portions of a work in progress and supplies the criteria.

4.2. Genetic CBR

Developing the training set is probably the most difficult aspect of employing CBR. As described in [8], it is often necessary to 'complete the input space': the net may create patterns which satisfy the criteria deduced from the training set, but which are not acceptable to the trainer. This indicates that the training set was inadequate. The offending patterns should be added to the training set, together with a corresponding 'this is not good' judgement.

In a *genetic* CBR implementation, the training set is produced incrementally by the net, and the user's role is only to evaluate successive creations. After a sufficient training set is accumulated, the net is trained, followed by accumulation of new examples, etc., until the creations are judged satisfactory by the user. (The name is suggested by a vague similarity to genetic optimization algorithms). We note that genetic CBR provides a reinforcement learning variation of

[1] Contact me if you are not convinced of this...

backpropagation [7].

5. Simulation

To illustrate the feasibility of attentional CBR in handling larger problems, we chose a simple problem from tonal music composition, specifically, to generate chord sequences. The (admittedly artificial) training set consisted of 80 random three-chord sequences, together with a quick rating along a traditional/nontraditional dimension. Additional examples were obtained while completing the input space (genetic CBR).

Chord sequences were constructed by this procedure: a I-V-I skeleton was used as a seed sequence. Chord pairs were then picked at random on the growing sequence and provided context for adding an intermediate chord by CBR. This process was repeated until sequences of a desired length were generated. Several such sequences are listed in Fig. 1. A more realistic problem, that of producing melodies using a training set derived from sources such as sight reading books, is being explored and results may be presented at the conference.

(0.200000 I III VII IV III V VII VI V VI VI I)
(0.200000 I II II VII V VII II VII VI III II I)
(0.500000 I IV I II V IV III IV VI II V I)
(0.500000 I V IV V IV V II VII VII III VII I)
(0.800000 I V IV I V V I VI IV V I I)
(0.800000 I I II V VI IV VI VII I I V I)

Fig. 1: Chord sequences generated by an attentional CBR trained on simplistic chord progression examples. The leading number indicates a conservative/unconstrained criterion (1=conservative).

References

1. Dorfman, D. and H. McKenna. 1966. "Pattern preference as a function of pattern uncertainty". *Canadian Journal of Psychology* 20: 143-153.
2. Duff, M. 1989. "Backpropagation and Bach's 5th Cello Suite (Abstract)". *International Joint Conference on Neural Networks.* Washington: IEEE, pg. II: 575.
3. Hiller, L. and L. Issacson. 1959. *Experimental Music.* New York: McGraw-Hill.
4. Jones, K. 1981. "Compositional applications of stochastic processes." *Computer Music Journal.* 5(2): 45-61.
5. Jordan, M. 1986. *Serial Order: A PDP Approach, TR-8604.* San Diego: University of California, Institute for Cognitive Science.
6. Lerdahl, F. and R. Jackendoff. 1983. *A Generative Theory of Tonal Music.* Cambridge Mass.: MIT Press.
8. Lewis, J., 1988. "Creation by Refinement: A creativity paradigm for gradient descent learning networks." *International Conference on Neural Networks* San Diego, pp. II: 229-233.
7. Lewis, J. 1989. "The Artificial Creativity Algorithm." *Submitted for publication.*
9. Park, K. 1988. "Sequential Learning: Observations on the Internal Code Generation Problem." *Unpublished paper.*
10. Rumelhart, D. and J. McClelland, Eds. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, Mass.: MIT Press.
11. Sejnowski, T. and C. Rosenberg. 1986. *NETtalk: A Parallel Network that Learns to Read Aloud.* EECS technical report EECS-86/01. Baltimore: Johns Hopkins.
12. Todd, P. 1988. "A Sequential Network Design for Musical Applications." *Proceedings of the 1988 Connectionist Models Summer School* Pittsburgh: Carnegie Mellon, pp. 76-84. (to be reprinted in *Computer Music Journal.* 13(4).)