A connectionist system for exploring melody space

Peter M. Todd

The Rowland Institute for Science
100 Cambridge Parkway, Cambridge, MA 02142
ptodd@sfo.rowland.org

ABSTRACT

Connectionist (neural network) systems allow a new approach to algorithmic music composition. We present here a connectionist system which allows composers to explore regions of "melody space" nearby other melodies of the composer's choosing. The composer selects a set of melodies that will define the melody space, positions them on a 2-d plane with a mouse-based graphic interface, trains a connectionist network to produce those melodies, and listens to the new "interpolated" melodies that the network generates corresponding to intermediate points in the 2-d plane. Future enhancements include using another network to automatically rate the interpolated melodies, allowing the composer to only listen to promising new creations.

1. INTRODUCTION

Connectionist or neural network methods are finding increasingly application in composer music (Todd and Loy 1991). Connectionist models use the parallel distributed processing abilities of a large number of simple processing units to replace the single central rule-following processor of the traditional von Neumann-style machine. Neural networks are typically used to learn mappings from inputs to outputs in a non-symbolic (or sub-symbolic) fashion, and the generalization abilities of the many interconnected neuron-like units acting in concert allow the network to map to new inputs in reasonable ways, producing appropriate outputs even in previously unseen situations (see Todd 1989, 1991 for more details, including how networks learn new behaviors by adjusting the weights on their connections between units). As a consequence, the connectionist paradigm for computation allows a distinctly different approach to algorithmic composition from that of standard symbolic AI. Rather than requiring the design and implementation of a complex rule-set for the step-by-step construction of new pieces of music, connectionist networks employ learning and generalization to discover "rules" and relationships directly from musical examples they are taught, and then use this newly learned knowledge to create new instances. In particular, a sequential network, in which the outputs are connected back to the inputs to form a recurrent dynamical system, can be trained to produce the next note in a given melody based on its memory of the previous ones; when given a new starting set of notes, it will generalize from what it has learned to produce a musical sequence following reasonably from that beginning. This system can be further extended by training it on more than just one melody, and then inducing it to combine aspects of the different melodies it has learned into one more-or-less seamless new whole.

In this paper, we describe a system being developed which encourages experimentation with such compositional blending in an interactive fashion. The composer can specify several input melodies which a network is to learn initially, each with a corresponding "location" in 2-d space. After the network has learned these melodies and their spatial associations, the composer can specify a new position in 2-d space which will form the basis of a new blended composition; all of those original melody locations which the new point is near will go into creating the new composition. Thus, if four original melodies were learned and associated with the corners of a square, a new melody corresponding to the center of the square would incorporate aspects of all four, while one positioned on the boundary edge of the square will mostly incorporate aspects of the two original melodies on the bottom two corners. By choosing locations and listening to their corresponding new compositions interactively, composers can quickly and easily explore a rich melodic space. In the next section, we briefly present the type of network used to learn and produce new melodies; in section 3, we describe the melody-space exploration system; and in section 4, we discuss further enhancements to increase this system's usefulness.
2. THE NETWORK

To explore a "melody space" mapped onto real (two-dimensional) space, we first need a network which can learn to produce melodies associated with particular 2-d locations. The recurrent network shown in Figure 1, and described in Todd (1989), can fulfill this task. This is a simple 3-layer Jordan-style sequential network, which maps an exponentially-decaying memory trace of the previous notes in a melody (in the Context units), along with an indication of what melody is currently being produced (fixed in the Plan units), through the "hidden units" of the middle layer which recategorize the inputs, to the next note at the output layer. Each output of the network represents the single pitch to be used next in the current generated monophonic melody (polyphonic schemes can also be implemented with more complex mechanisms). Durations of the individual notes can be handled in several ways, including a "time-slice" representation (used in Todd 1989), or with separate duration-indicating output units. (Duration is omitted from Figure 1 for simplicity.) All of the units have real-valued activations in the range from 0.0 to 1.0.

This type of network has been used in previous studies to learn and produce a wide range of simple monophonic melodies, and one network can be trained to produce several melodies through the use of different plan-unit inputs that identify each melody. With two plan units, an x-y coordinate (a pair of real values from 0-1) can be used as the plan, associating each melody learned and produced with a point in the 2-d unit square. To train the network to produce a certain melody associated with a certain x-y position, the x-y values are clamped onto the plan units, the context units are started out empty (all 0), activity is propagated forward through the network to the output units, the current output is compared to the desired output, and the difference (error) is used to adjust the weights in the network. Then the target values are passed back along the recurrent connections to be stored in the decaying context representation (this speeds up learning compared to using the actual output values as the context during training), and activity is propagated forward again, and the next outputs are compared to the next target pitch. This is repeated for the entire melody in order, and is continued for each melody to be learned until the network can produce them all without mistake. To generate new melodies, a new x-y plan is clamped onto the plan units, activity is propagated through the network, the output is computed and saved as the first note of the new melody, and that actual output (rather than a target value, since there is no target for new melodies) is passed back along the recurrent connections to be stored in the context. This cycle is repeated until a new melody of desired length has been built up. (The actual learning rule used is slightly different from that of Todd, 1989: in this case, rather than the summed-square error term used previously, we employ a multimodal-based training scheme, which assumes that only one output unit is on at a time, matching our current network behavior.)

---

**Figure 1.** The recurrent sequential network used to learn and generate melodies.

**Figure 2.** a. 2-d melody space shown with five positioned melodies. b. Hypothetical equal-melody contours shown around the five melodies from a.
3. THE COMPOSITIONAL EXPLORATION SYSTEM

With our trainable melody-generating network in hand, we can proceed to the workings of the melody-space exploration system. This system is based on a mouse-driven graphical interface running under X11. To use this system, the composer first collects a small set of monophonic melodies which will define the melody space. These melodies (which can be many measures long) are entered into a file in a simple alphanumeric representation before the system is started up. The first task within the system is to assign each initial melody to a location in 2-d space. This is achieved simply by positioning a number corresponding to each melody at the desired location with the mouse. The melodies can be laid down in the space in any pattern desired—regularly spaced around a circle, in a clump, on a grid, at random, etc. A simple positioning pattern for five melodies is shown in Figure 2a.

Once the final positions for the melodies have been chosen, network training begins. Training is the time-consuming portion of this process, and depending on the machine used, can take overnight or longer. Training time also depends on the number and length of melodies used to specify the melody space. As mentioned earlier, training usually ends when the network can reproduce all the original melodies without error; however, the composer can stop training at any time before that and explore the incompletely-specified space at that point, searching for interesting melodies that may already have emerged. To explore the constructed melody space after or during training, the composer simply clicks on a position in the displayed 2-d space, and a melody corresponding to that location is played immediately (either through internal speakers or via MIDI). The generation of the new melody occurs very rapidly, because all that has to happen is to clamp the selected x-y coordinate values to the new network plan, clear out the context units, and start sequentially propagating activation around the network and play the successive outputs. This click-and-listen process of exploring the melody space can continue as long as the composer is interested, and the promising melodies found along the way can be saved for future use. Additionally, the newly-generated melodies (or others of the composer's choosing) can be added to the training set at any time (along with x-y locations to associate them with), and the network can be further trained to create a more or less different melody space to explore.

4. EXPLANATIONS AND ENHANCEMENTS

In its basic form, this is all there is to the melody-space exploration system. This method works to create new melodies similar to the initially chosen ones, combining aspects of melodies nearby in the 2-d space, because the learning and generalization behavior of the sequential network, operating as a nonlinear dynamical system, essentially constructs a landscape of basins and attractors around the original 2-d locations used in training. That is, the constructed melody space can be visualized as a 2-d surface above the 2-d plane, with a deep valley (a basin of attraction) centered at each original melody position, and with ridges, plateaus, and peaks between those positions. The closer the composer chooses a new point to one of the valleys, the more the newly-generated melody will be like one of the originals; the "higher-up" the landscape a point is selected, the more the new melody will be a different combination of parts of the original melodies.

However, not every different x-y point in the 2-d plane will generate a different new melody; in fact, there are typically rather large areas where different places will produce the same output melody (see Teddi 1989 for examples in the 1-d case). Thus the composer can waste a lot of time hearing the same melody over and over at different points. To alleviate this problem, the first enhancement we are developing is a "contour-mapping" module which will draw contour lines on the 2-d space that show the boundaries between regions where different melodies are generated. A hypothetical such contour map for the space trained in Figure 2a is shown in Figure 2b. These contours clearly show the valleys, ridges, etc., formed in the melody-space landscape during training, and they will make the system much easier to use, by letting the composer only sample the space where different melodies can be found. We also plan to show these contour lines forming and shifting during the course of training, to help the composer decide when to stop training and have a look at what's been constructed so far. (The construction of the contour lines will take some time as well, depending on the resolution chosen for determining them, since the melodies generated at a large number of x-y points will have to be compared for each map.)
With this consorted landscape image in mind, we can devise several other ways to modify the melody space. First, to increase the size of a melody’s basin of attraction, we can train the network to produce that melody with noisy (randomly modified) plan values, specifying a small neighborhood in melody space, rather than a single point. Second, we can use several plan-locations for the same melody, to create a ridge (if the locations are in a line) or plateau (in a cluster or polygon) where that melody will be produced. Third, we can sharpen up the contours in the melody space by training the network longer; the bigger the weights become, the more discriminations between locations the network is likely to make, creating “steeper” slopes in the landscape, and hence more contours and different melodies to explore.

Even with the contour lines indicating where to sample the space, though, the composer may be faced with a lot of new melodies to search through and listen to. Our system could be further enhanced with the addition of an indication of just how promising the new melodies are. Thus, if we could color-code the different regions of the space, going from red for unpromising melodies to green for possible hits, the composer could choose to just listen to those melodies most likely to be of interest. Rating the new melodies in this way is obviously an unachievable goal, since coding “goodness” or “promise” of melodies is completely subjective, but we can make a tentative start in this direction with the addition of a network trained to produce single-valued judgments of melodies. This network could use as inputs the entire melody presented at once (as in the rating networks of Lewis 1991), or each note presented sequentially (using a modification of the network presented here, with activity summarized at a single rating output unit). The network’s output would be a rating of the “goodness” of the melody on some dimension, from aesthetic pleasantness to simple tonality; the exact rating function used could be selected by the composer, either from a predetermined list of possibilities, or through ongoing training. In the latter case, each time the composer played a melody from the system, that melody would be used as input to the rating network, the composer would provide a rating judgment to be used as the network’s target output, and this input-output pair would be used to train the network to provide more useful ratings. In this way, the composer has to listen to and subjectively rate a number of melodies during the exploration of melody space anyhow, we can try to capture some of that expertise in a network. The single output value of the rating network would be converted into a color using some code, which would be displayed for the entire region occupied by that melody. (We can perhaps add further color to this notion of rating judgments by employing Jacobs and Jordan’s 1991 collection of expert network modules, each specialized for a different musical style, or Jordan and Rumelhart’s 1991 forward modeling idea for using a rating network to actually improve the learning of the composing network.)

Finally, we may like the network to produce the kind of results we’d expect, as human music listeners—that is, we’d like the melodicist that it produces nearby in melody space to actually sound similar to us. To achieve closer approximation to human expectations, we can use human similarity judgments for simple melodies both to adjust where the original melodies are placed in the melody space, and to constrain the formation of that space to match those judgments. This approach would require gathering a lot of experimental data from human subjects, and this would be time-consuming and expensive, but even the introduction of a little such data could help the networks between contour lines perform better. And the more human musical expertise we can incorporate into the composing (and rating) network, the more useful a tool our connectionist melody-space exploration system will become.

5. REFERENCES


