The Self-Supervising Machine

Benjamin D. Smith
University of Illinois at Urbana-Champaign
School of Music
Urbana, Illinois
bdsmith3@illinois.edu

Guy E. Garnett
University of Illinois at Urbana-Champaign
eDream, Illinois Informatics Institute
Urbana, Illinois
garnett@illinois.edu

ABSTRACT

Supervised machine learning enables complex many-to-many mappings and control schemes needed in interactive performance systems. One of the persistent problems in these applications is generating, identifying and choosing input output pairings for training. This poses problems of scope (limiting the realm of potential control inputs), effort (requiring significant pre-performance training time), and cognitive load (forcing the performer to learn and remember the control areas). We discuss the creation and implementation of an automatic "supervisor," using unsupervised machine learning algorithms to train a supervised neural network on the fly. This hierarchical arrangement enables network training in real time based on the musical or gestural control inputs employed in a performance, aiming at freeing the performer to operate in a creative, intuitive realm, making the machine control transparent and automatic. Three implementations of this *self supervised* model driven by iPod, iPad, and acoustic violin are described.

Keywords

NIME, machine learning, interactive computer music, machine listening, improvisation, adaptive resonance theory

1. INTRODUCTION

Machine learning (ML) continues to gain increasing application in the performing arts as the problems of interactive system control in live performance become more and more approachable and better understood [7]. The promise of the transparent union of live performer and complex multimedia performance is alluring, and with the development of on-line ML algorithms and the computing power required to run them in real time, such performances are rapidly becoming reality. Yet the extensive pre-performance training required of most musical ML applications poses a particular problem to the improvising musician who wishes to privilege spontaneous musical creativity during a performance.

We describe herein the design of a unique self-supervised system that employs unsupervised learning algorithms, specifically Adaptive Resonance Theory (ART), to automatically parse input music and gesture streams, locate significant feature areas, and train many-to-many mappings in real time. The result is an interactive multi-media system that generates mappings uniquely for each performance, extracting the particulars of a given input stream and creating a control space that produces rapid, tightly coupled responses.

2. MOTIVATION

The applicability of ML methods to problems in interactive musical performance is evidenced by the number and variety of applications and cases (see for example [4, 14, 16]). Recent systems, such as the work of Fiebrink et al. [7], focus both on real-time training, in order to better match the musician's work process, as well as the use of ML to discover new musical expressions. Rather than attempt to exactly duplicate preconceived mappings they encourage the exploration of unexpected results stemming from active training during a performance.

However, supervised ML implementations conventionally require that the musician define both their input material (i.e. what the system will learn to identify) as well as the desired outputs (the intended results in an interactive performance system) in advance of their use during a performance. Training the computer to produce desired outputs for given inputs serves to effectively build a complex computer music instrument driven by dynamic gestural controllers or acoustic instruments. This may be accomplished transparently during a performance [7], but requires extensive awareness and expertise on the part of the performer.

For example, consider a musician who, during an improvisation, trains the system to recognize two distinct melodic patterns, tying these to two different system outputs. As the piece progresses, the musician must anticipate their own movement to new melodic areas, retraining the system at each point. Failing this, the performer's connection to the computer becomes challenged, as the content of the music, i.e. the relationships between notes, events, and phrases, moves away from the domain that the system was trained for. This will be especially apparent in systems designed for discrete classification, i.e. where the system only produces outputs when a known input is observed, but is also problematic with interpolating systems due to the input's movement away from the known feature domain. This can be partially obviated by providing a sufficiently broad range of inputs to train on, but this requires a very substantial effort in defining or predefining suitable training sets.

Thus these systems can be prohibitive for an improvising musician who wishes to privilege spontaneity and creativity in the moment of the performance. Additionally, existing systems typically treat all categories equally, missing the musical interest resident in the inter-input relationships. We mitigate this dependence on predetermination of scope, by designing a system that automatically identifies movement to new areas of the input set (i.e. melody, in this example), and creates mappings to account for these per-

ceptive, musical developments.

These interactively determined relationships are better captured by unsupervised, on-line ML models, though the latter have seen virtually no application in interactive performance to date. These models allow the algorithm to discover classifications and find groupings and patterns across inputs based on relationships inherent in the data, rather than training on preconceived knowledge of the inputs (such as [1, 6, 10] applied to problems in Music Information Retrieval). Effectively, the computer is allowed to build its own interpretation of the musical work, listening in a fashion analogous to the human listener [3]. The primary problem posed by unsupervised methods is the potential for input to be categorized in ways unanticipated by the performer (however, this is also a possibility when playing with other humans). Inherently in "unsupervised" algorithms, the inputs are automatically parsed and categorized without human oversight or labels.

The capability of unsupervised learning to analyze musical material is shown by Gjerdingen [8] and Piat [12], who employ ART models to produce automatic classifications. The former found that the machine could automatically learn to identify formal changes in early Mozart compositions, discovering relationships without human supervision. Piat used a similar system to train a computer to hear the relative difference between "consonant" and "dissonant" music, comparable to human test subjects with surprising fidelity. However, neither of these projects ran in real-time.

Our ART implementation is faithful to [2], although our work appears to be the first real-time, performance oriented application of ART in music.

3. DESIGN

The design of our system consists of two primary modules, a *supervisor* component and the mapping network. The supervisor has an ART network at its core, itself running without supervision, examining the input feature stream for pitch-focal areas (in the case of musical input), and producing categorizations. The mapping is accomplished through a MLP network, enabling non-linear translation of inputs to outputs. The outputs must be defined in advance, as in [4, 7, 14, 16].

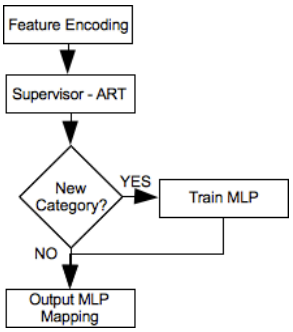

Figure 1: System ML interaction.

The connection between the two networks is unidirectional, with the supervisor acting as a trigger to retrain the MLP. Given a stream of feature data the ART will classify and group the inputs, creating categories of varying size that parse the total feature space. When a category is created, and the overall system decides it is a significant category, it is tied to a predetermined output and a training session commences. In the current implementations the realm of possible outputs is defined by a data set created in advance of operation. This is generated effectively through a pre-performance *improvisation* wherein the composer/performer chooses the domain of system outputs and orders them sequentially. As the performance unfolds the output set is traversed in order, creating mappings to the live inputs. The training session runs in the system's free time between input presentations, allowing the performer to continue uninterrupted by the training.

The system as described is currently embedded in three distinct implementations, driving audio synthesis and video animation with mobile touch devices (Apple iPod and iPad), as well as acoustic instruments (a violin).

3.1 Supervisor: ART

The supervisor models both a short-term memory (retaining the last several seconds of input) and long-term memory (grouping, relating, and retaining inputs for later recall) based on contemporary theories of human audition and perception [11, 15]. This is accomplished as two distinct components: a feature encoding module (the short-term memory, STM) and the unsupervised ART implementation (the long-term memory). The aim is to provide the computer with a method of parsing the feature inputs that mimics a human's abilities, such that a distinctive, to a human, change in the inputs (and in the originating music or gestures) is recognizable by the machine.

3.1.1 Short-Term Memory

The feature encoding can be accomplished in a number of ways, dependent on the nature of the input data. Our implementations employ two distinct models. The first is a simple scaling and grouping of independent values representing significant components of the input data. In the case of the mobile touch implementations this involves tracking the touch position(s), touch velocity (delta between touch samples), and touch curvature (degree of change in direction between velocity samples proportional to distance traveled), and transforming these values into a vector where each element (x) obeys: $0 \leq x \leq 1$.

The same model is similarly employed for part of the analysis of the acoustic violin input, creating a feature vector from spectral centroid ("center of mass" of the spectrum, or perceptual "brightness" [13]), spectral noisiness (how tone-like or noise-like the sound is), and register (average pitch over a two second window).

Spatial encoding [5, 8] comprises the other primary STM model, enabling the transformation of melodic and pitch sequences into feature data that the ART can process. A simple neural network with attenuated feedback forms its core, with one node for each unique token in the potential input set (this model is commonly used in natural language processing where each character in an alphabet is used as a token). When a token is presented to the network the corresponding node is fully activated and the network produces an output (x). This output is in turn fed back into the network, attenuated by a small amount (α) (typically set to retain five to nine inputs in the network [9]):

$$x_t = \alpha x_{t-1} \tag{1}$$

Thus the occurrence of each token in time is transformed into a vector, where each element indicates relatively how recently that token appeared. The vector can then be understood as a spatial representation of a point in the sequence, creating a position and movement within the token space. For musical input, pitch classes and interval classes (based on the twelve-tone equal-tempered tuning system) become ready token sets, creating feature vectors that depict melodic and harmonic movement and allow the detection of motivic and pitch-class set relationships (see [8]).

3.1.2 Long-Term Memory

Once the STM is created, the resulting vectors are fed to the ART module for classification. The ART is a competitive neural-network using unsupervised training, creating feature categories based on an ordered sequence of input vectors. That the input is ordered is significant, as different orderings of the same data will produce divergent classifications. While usually considered a limitation, this is an asset in music parsing where the order is carefully contrived by the artist and is important—and usually specific—to the particular musical work. Just as a human listener relates later melodic development (such as a sonata-form development section, or recapitulation) to earlier auditions (i.e. the exposition), so does the ART algorithm.

When presented with a new input vector ($\boldsymbol{I}$) the ART algorithm first obtains a resonance measure (T) through the comparison of each known category ($\boldsymbol{w}$) with the new input (Eq. 2).

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\gamma + |\mathbf{w}_j|} \tag{2}$$

For a given input ($\boldsymbol{I}$) the resonance measure is calculated with choice function (T), comparing the input with the adaptive weights ($\boldsymbol{w}$) of each category (j). A choice parameter (γ) affects the matching of inputs to the closest subset category, and is typically set close to 0 to achieve this. The "fuzzy AND" operator $\wedge$ is defined by

$$(\mathbf{x} \wedge \mathbf{y}) = min(x_i, y_i) \tag{3}$$

and the norm $|\bullet|$ is the L1 norm

$$|\mathbf{x}| = \sum_{i=1} |x_i| \tag{4}$$

Before learning ensues the strongest resonating node must pass a "vigilance" test, to ensure it remains within a preset limit (Eq. 5). If the node's size is acceptable then it is selected and allowed to learn based on the input. On the other hand if by incorporating the new input the category size (in feature space) would increase beyond this limit (the "vigilance" parameter, p, in Eq. 5), then this node is rejected for this iteration and the next most resonant node is considered.

$$\frac{|\mathbf{I} \wedge \mathbf{w_j}|}{|\mathbf{I}|} < p \tag{5}$$

The rejection of all existing category nodes results in the creation and training of a new category node. The details of the ART algorithm we employ are described at greater length by Carpenter et al. [2].

The other control parameter of significance is the "learning rate" of the ART network. This parameter allows the network to both train new inputs immediately and still adapt slowly, retaining the identity of older categories. Setting the learning rate high causes categories to expand and fully incorporate new inputs while setting it low causes the categories to adjust slowly, settling into an average area of the feature space. The implementations below set the learning rate near 1, allowing identified categories to adapt to new inputs immediately, ensuring reproducible classifications (setting the learning rate low can cause subsequent presentations of the same inputs to be classified differently, dependent on category expansion rates).

3.2 Mapping: MLP

The MLP is a feedforward neural-network, consisting of multiple layers of nodes fully connected in a directed graph, which maps sets of input data onto sets of output data. The input nodes have a simple linear activation function but the nodes of the hidden internal layers have non-linear activation functions (typically, and in our case, these are sigmoids). Backpropagation is used to train the network, repeating iteratively over the course of a performance session.

Training of the MLP occurs frequently but unobtrusively to the user as new data is created during a performance. The ART module generates paired sets of inputs and outputs, which are updated based on musical (or gestural) developments as the work unfolds. For every updated training set a training period is initiated wherein the inputs and outputs are presented to the backpropagation algorithm iteratively until an error function falls below a given threshold or a safety time-out is reached. For the error function we employ a simple distance measure of the amount of correction the network undergoes for each new input set (i.e. an average of how much each node's parameters were changed during the training iteration). When the error condition is met (typically, the average change is below 10^{-5}) the training session ceases. The safety time-out is triggered when the error condition fails to move below a higher threshold (typically 0.01) within 20,000 iterations. These thresholds were chosen by trial and error over many tests where we have observed that the configuration of the MLP (i.e. the number of hidden layers and the flatness of the activation functions) is the primary determinant in the convergence of the training periods. Once properly configured for the task the MLP trains and converges very reliably.

During training, the MLP continues to map inputs to outputs in an uninterrupted fashion, although the mappings produced during a training cycle can be unpredictable. For the initial training period, when the network goes from a randomly initialized state through the first training set, the mappings quickly move from random (but repeatable) to trained and expected. Subsequent retraining periods produce significantly less variation as the network has shorter distances to go. The typical training period comprises several thousand iterations spanning between 0.5 and 2 seconds, and proves barely noticeable to most users. At all other times the network operates as anticipated, efficiently transforming inputs into control outputs (where inputs are received approximately twenty times a second, in our applications).

4. APPLICATIONS

4.1 iPad 1

The simplest interface and implementation consists of an iPad driving a granular synthesis engine. Here we take the two-dimensional touch input (position on the surface) as the feature data, scaling it and feeding it directly to the ART module. This serves to parse the area of the touch screen uniquely for each session and makes a simple and dramatic demonstration of the system. Every time the ART identifies a new region of the feature data (and thus the screen) it is mapped to a new output from the preset granular synthesis outputs. The result is a two dimensional mapping of the nine-dimensional granular synthesis parameters, where any touch on the screen is mapped to a point in the synthesis space.

4.2 iPad 2

We attempt to characterize *drawing style* in the second application, analyzing the iPad touch data for speed of movement, curvature of the line, average direction of movement (taken over sixteen samples) as well as simple position. This results in five parameters that are now mapped through the MLP to the nine-dimensional granular synthesis parameter

space. In this application the user is able to achieve much finer control of the output, mapping many more input areas to outputs. Thus, for example, a straight, slow line in the center of the screen moving to a straight, slow line near the edge produces a gradual sonic change, while a change to a curvy, fast line and back to straight traverses the output parameter space much more dramatically.

4.3 Violin

Providing control to an acoustic musician requires many more dimensions of input. This application utilizes analyzed parameters of the sound (brightness, noisiness, amplitude, and average frequency), as well as musical and textural components (rate of attack and pitch class). Both simple scaling mechanisms and spatial encoding algorithms are employed to produce two separate feature vectors (one for immediate sound parameters and one for a pitch-based short-term memory), which in turn are fed to two distinct ART modules. The outputs of the ARTs serve to train two different MLP networks, one driving the granular synthesis engine and the other controlling real-time graphical animations.

5. CONCLUSIONS

This self-supervising model shows the applicability of unsupervised and supervised ML algorithms working together in an interactive multi-media performance system. All three of our implementations have been employed successfully in demonstrations and performances, and additional testing and development is underway. While the system promises full autonomy for the interactive computer it is currently limited by the pre-definition of the outputs. In this way it functions like supervised ML systems of recent decades. However this may be alleviated by providing the system with methods to analyze and filter the outputs. The parsing of the outputs can be done in a fashion analogous to the inputs, and similarity measures (i.e. various distances between features and categories) can give the computer a path to relating inputs and outputs automatically. Thus a minimal shift in the input music or gestures can be matched with a minimal change in the multi-media output, and significant movements can be treated similarly.

In the current implementations the level of detail set in the ART module has a great effect on the results of the mapping network. When set to produce more general categories (accomplished through a lower vigilance setting) training points are created very far apart in the input feature space. When set to be more precise the training points are put close together. The former gives the performer more gradations of control, but requires large musical shifts to produce noticeable changes in the multi-media system. On the other extreme the smallest changes (changing notes or slight dynamic levels) causes significant changes in the output. Finding a suitable middle ground requires a period of testing to tailor the ART parameters to the music that a given performer desires to play.

While the goal of a fully automatic self supervising system is approaching, the current implementations still demand a noticeable amount of awareness on the part of the user. Although the performer is free to improvise and continually explore new material the mappings will continue to appear new as well. Our experience has shown that this lack of constraints can be challenging. The responsibility is entirely on the performer to remember what they presented to the system and affect its recreation if they desire a return of the same multi-media outputs.

6. REFERENCES

[1] J. Aucouturier and F. Pachet. Tools and architecture for the evaluation of similarity measures: case study of timbre similarity. *Journal of the American Society for Information*, Special Issue on Music Information Recrieval, 2004.

[2] G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

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

[4] R. B. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. In *Proc. International Computer Music Conference*, pages 344–347, 1997.

[5] C. J. Davis and J. S. Bowers. Contrasting five different theories of letter position coding: Evidence from orthographic similarity effects. *Journal of Experimental Psychology: Human Perception and Performance*, 32(3):535–557, 2006.

[6] P. P. de León and J. Inesta. Pattern recognition approach for music style identification using shallow statistical descriptors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(2):248–257, Feb. 2007.

[7] R. Fiebrink, P. R. Cook, and D. Trueman. Play-along mapping of musical controllers. In *Proceedings of the International Computer Music Conference*, 2009.

[8] R. O. Gjerdingen. Categorization of musical patterns by self-organizing neuronlike networks. *Musical Perception*, 1990.

[9] A. Miller George. The magical number seven, plus or minus two: some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[10] F. Pachet. Musical data mining for electronic music distribution. *Web Delivering of Music, 2001. Proceedings. First International Conference on*, pages 101–106, Nov. 2001.

[11] I. Peretz and R. J. Zatorre. Brain organization for music processing. *Annual Reviews*, Psychology(56):89–114, 2005.

[12] F. G. P. Piat. *Artist: Adaptive resonance theory to internalize the structure of tonality*. PhD in human development and communication sciences, University of Texas, Dallas, Aug. 1999.

[13] E. Schubert, J. Wolfe, and A. Tarnopolsky. Spectral centroid and timbre in complex, multiple instrumental textures. In *Proceedings of the 8th International Conference on Music Perception and Cognition*, Sydney, Australia, 2004. University of New South Wales.

[14] B. Thom. Interactive improvisational music companionship: a user-modeling approach. *User Modeling and User-Adapted Interaction*, 13:133–177, 2003.

[15] B. Tillmann. Music cognition: Learning, perception, expectations. In *Computer Music Modeling and Retrieval. Sense of Sounds.*, pages 11–33. Springer, Berlin, 2008.

[16] D. Wessel. Connectionist models for musical control of nonlinear dynamical systems. *The Journal of the Acoustical Society of America*, 92(4):2402, Oct. 1992.