ML.* MACHINE LEARNING LIBRARY AS A MUSICAL PARTNER IN THE COMPUTER-ACOUSTIC COMPOSITION FLIGHT

Benjamin D. Smith **W. Scott Deal**
Indiana University Purdue University Indianapolis, Department of Music and Arts Technology
bds6@iupui.edu, deal@iupui.edu

ABSTRACT

This paper presents an application and extension of the ml.* library, implementing machine learning (ML) models to facilitate "creative" interactions between musician and machine. The objective behind the work is to effectuate a musical "virtual partner" capable of creation in a range of musical scenarios that encompass composition, improvisation, studio, and live concert performance. An overview of the piece, Flights, used to test the musical range of the application is given, followed by a description of the development rationale for the project. Its contribution to the aesthetic quality of the human musical process is discussed.

1. INTRODUCTION

Machine learning algorithms of broad variety are firmly established in research that integrates them with musical functions and processes. For example, algorithmic tools developed by Cope [1] explore musical counterpoint. Raphael articulates a system employing algorithms for musical accompaniment [2]. Melo, Drevert, and Wiggins [3] study machine learning processes for sound diffusion performance. George Lewis' *Voyager* analyzes an improvisational performance and then generates responses in real-time [4]. .Weinberg and Driscoll [5] applied machine learning to Haile, a music robot designed for spontaneous acoustic musical performance through interaction with a human musician. *Triple Point*, led by Pauline Oliveros, is a computer-acoustic group that employs FILTER, a real-time algorithmic improvising partner [6]. In light of this work, particularly in the improvisational, spontaneous compositional sphere, the authors present the *ml.** application as a tool for real-time algorithmic musical collaboration. Building on earlier work that utilized *ml.** for harmonic and melodic content through the MIDI protocol [7], this effort employs algorithms to analyze analog pitch and rhythmic material, and then generate the same audio sounds, now sampled, as sonic material for computer-composed, real-time responses.

The application is intended to satisfy the following criteria.

1. Interfaces easily with non-technical musicians, requiring a minimum of specific knowledge.
2. Analyzes and outputs melody, harmony, rhythm, tempo, timbre and dynamics.
3. Generates musically sophisticated and nuanced responses with minimal extra-musical manipulation by the user/musician.

In a sense this work represents a step in the continuing search to create an intelligent musical partner, capable of improvising and performing as a musical, expressive partner and equal in concert with human musicians.

One of the particular problems of implementing such systems lies in the difficulty of integrating the mechanics of ML into the aesthetics of music [8]. In this work we developed a ML system that helps musicians augment their performance while minimizing additional cognitive load required to interact with the system. In turn, the involvement of a human performer then helps guide the aesthetic relevance of the ML system.

Flights (Deal, 2014) was composed for this project in mind. This approach gave insight into how closely artistic expectations can be matched by the chosen design. The ML interface was constructed by incorporating *ml.**, the *Machine Learning Toolkit for Max 5+* [7].

2. FLIGHTS

Flights is a work that creates a dynamic musical environment through the integration of rhythmic and harmonic movement with a machine learning application. The structure is a series of delineated sections comprised of sets of interchangeable rhythmic structures (see figure 1) and chord progressions (figure 2), fashioned so the performer has ample freedom to engage and interact with the output of the *ml* application. The performers work their way through the harmonic material by interspersing virtuosic passages with space for the *ml* application to fill out. Any treatment of the work is feasible, and extends to an additional performer on the score and also the implementation of live digital signal processing and media exploitation. If performed as an instrumental duet, each player should play soloistically yet in proximity and aesthetic relationship to the other, and both should send their musical input the same *ml* application (acting as a third

performer). Grouping instrumentalists with previously mentioned elements, *ml* and live processing, shapes the aural nature of a performance space through the arrangement of performers and loudspeakers. While each section of the work is performed soloistically by the instrumentalist(s), *ml** is capturing harmonic and rhythmic data, categorizing it, then outputting a derived response, yielding cascades of re-supposed harmonic, rhythmic, and melissmatic material that emanates throughout the space, resulting in a series of dynamic musical episodes. *Flight* is designed for performance in either a single physical space, or distributed telematically between multiple sites over the Internet. The open score suggests instruments that include percussion, piano, harp, strings, guitars, woodwinds, prepared/augmented instruments, controllers, and computer interactivity. *Ml** is introduced into the design of the work via the structural shape of the composition, in which sections consist of virtuosic passages followed by large rests, which in turn create room for liberal amounts of interactivity. The success of a performance lies in the timing, placement, and juxtaposition of virtual and live sound together with the interaction of instrumentalists and *ml**.

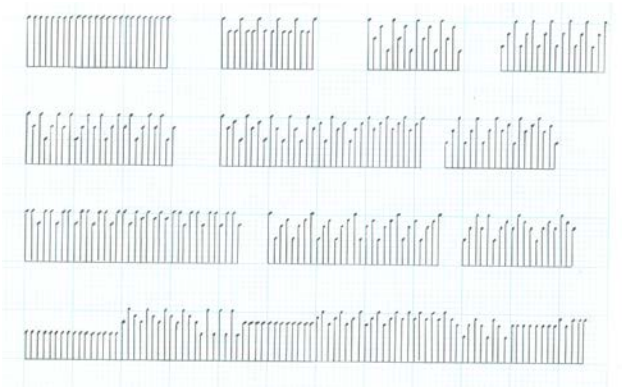

Figure 1: Rhythmic structures for guiding improvisation in *Flight*.

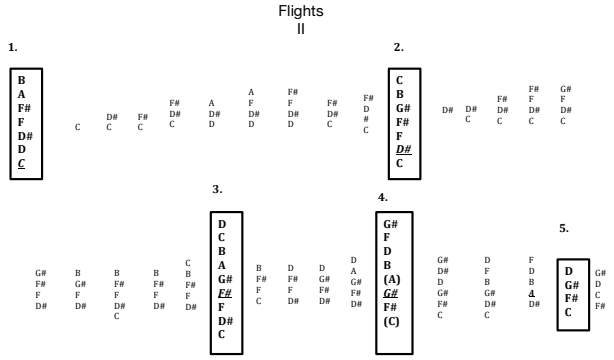

Figure 2: Pitch set progressions for *Flight.*

3. DESIGN

3.1 Objectives

The primary design goals are to create a system that can produce and interact with sufficient musical diversity and richness, yet require a minimum of extra-musical communication. That is, the design intentionally avoids the use of non-musical signals or communication (such as button presses or a trigger pedal), instead aiming to respond to the human participants in a manner analogous to other human performers, understanding musical gestures and signals in an attempt to derive the human's intent. Ideally this will allow trained musicians to begin playing and interacting with the system, leveraging their musical experience and diminishing new, specific learning required to play with this computer system.

A secondary goal for the design of the system is to portray aspects of creativity, encouraging expressive interaction and musical production. Providing sufficient flexibility, diversity, and possibility of the generative musical content in order to appear creative is a complex task for deterministic systems. Indeed, systems relying on pre-set algorithms are confined by finite bounds and may never reach the subtle complexity of human expression. Stochastic processes are typically employed to expand these possibilities. While randomness may accidentally uncover new musical ideas and spark creativity and interest in a human user, the system itself is wandering blindly, just as likely to produce musical noise as it is to provide stimulating material.

Machine Learning models provide a ready alternative in order to give the computational system some understanding and knowledge of the domain it is working in. These models are seeing increased use and exploration in computer music today as their potential begins to be explored in many directions [8, 9].

ML techniques fall broadly into two categories: supervised–models that require a complete dataset in advance and typically involve careful tuning and selection by human users, and unsupervised–models that work in an adaptive fashion and provide minimal controls to the human user. The former category provides advantages when a full dataset is available, such as in processing a musical work for automated analysis, and when the user has specific advanced knowledge about the nature of the data (i.e. that the work employs a functional harmony model, for example.) The second category, unsupervised techniques, provides more exploratory approaches, wherein the models discover their own correlations and construct their own patterns to enable recall and processing [9].

A mixed model indicates the possibility of taking advantage of the strengths of both approaches. This "self-supervised" notion [10] affords the system the opportunity to identify salient patterns in the data-stream (i.e. musical input data from a partner musician) and use this knowledge to train and re-train supervised models. This process is analogous to [8, 10] but further automates the system, giving it the ability to make suppositions about the musician's intent and encode complex and varied relationships.

3.2 Design

The system consists, at a high level, of the elements shown in Figure 3. After audio from the human partner is digitized it passes through a feature encoder/extractor to provide relevant information for analysis. This feature

data is used as a look-up query 'word' to find the closest match in the system's memory. This match is then output to a decision module that uses the identified word to inform musical playback. Working in parallel, a self-organizing map is used to extract closely related 'words,' enabling a controllable range of responses.

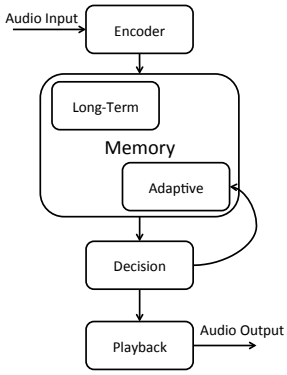

Figure 3: System diagram for Flight computational agent.

3.2.1 Encoder

In order to be ultimately flexible the system's audio feature extraction must capture input with sufficient specificity, yet in a form general enough to allow a diversity of instruments, play styles, and musical genres to be characterized adequately. Towards this end the encoding relies on a "chroma" analyser [11]. This reduction bins all the analysed frequencies (from a standard FFT) according to the 12 chromatic pitches of typical western just tuning. Thus all frequencies that are in an octave relationship where C is in bin 0, all C#s are in bin 1, etc. This gives a 12 element vector characterizing the harmonic content of a given sample window.

In addition to the chroma features, the system appends measures of brightness, noisiness, and loudness [12]. These are based on perceptual models attempting to capture salient aspects of the input sound.

The system stores all inputs in an uncompressed database of audio files, along with the analysed feature vectors associated with the audio. Analysis can be set to occur at any rate lower than 10hz (an upper limitation of the chroma implementation employed). Features are typically extracted using overlapping windows, effectively doubling or quadrupling the feature vector rate. Testing typically employed a two-fold window at 1hz, producing a feature vector every 500ms.

3.2.2 Memory

The memory of the system comprises two parts, a fully lossless 'long-term' memory, retaining all audio inputs and analysed features, and an adaptive 'short-term' memory. The size of the long-term database, after operation for any practical amount of time, is not insignificant and an efficient sort and search algorithm is required for real-time performance. This design uses a k-d tree model implemented as a Max external, extending the ml.* library.

The k-d tree [13] is a space partitioning data structure for organizing points in a k-dimensions. The k-d tree is particularly useful in retrieval and searches involving multidimensional search keys, as in the present case. Here, each feature is one dimension in the data space and the k-d tree serves as a sorting algorithm providing efficient nearest neighbour search and retrieval (with a performance of $O(\log n)$).

Training of the k-d tree can occur at any point but typically takes place at session boundaries, i.e. when the user has provided sufficient new audio to require re-training. Retraining operations perform on the order of milliseconds, however the restructuring of the data can result in new tendencies in the system (i.e. apparent proclivities to make certain musical selections as a result of the new tree structure), and seems more easily accepted by the musician between play sessions, rather than in mid-performance.

During performance every feature vector is used as a search query into the k-d tree to locate the closest trained exemplar of that vector. This provides the system with a core function: the ability to match the human as closely as possible and play in a unison fashion with them.

However, this ability to mimic on its own is highly restrictive and limited. To enhance the decision making aspects and apparent creativity of the system an adaptive short-term memory is incorporated, employing a machine learning model, a self-organizing map, to allow the analysis and rapid recollection of relationships between feature sets.

The self-organizing map (SOM) [10, 14] provides unsupervised clustering and classification, mapping high-dimensional input data onto a two-dimensional output space, preserving the topological relationships between the input data items as faithfully as possible. The primary strength of the SOM is its fundamentally visual metaphor, translating higher dimensional data into an easily portrayed map. In other words, the SOM produces a projection of the input data space onto a two-dimensional map such that proximity on the map parallels some sort of similarity (or proximity) in the source data space. It is a computationally cheap model, and arguably mimics human cognitive models leading to results that parallel human perception and decisions at a basic level.

At its core the SOM is a neural network lattice of nodes connected in a two-dimensional configuration (although higher dimensional arrangements are possible) in which each node represents a possible category in the input space. The SOM may also be considered a nonlinear generalization of principle component analysis over which the SOM arguably provides many advantages [15].

In this system the SOM is presented with inputs, and a search is performed to locate the most similar (i.e. closest, using a Euclidian distance measure) node in the map. Learning is performed continuously, adapting the winning node and its neighbors to more appropriately represent the new inputs. The learning is calculated as a gradual reduction of the distance between the input and the matching map node (and the amount of change is used as a control signal for the Decision module, below). The result of adapting nodes in a gradually decreasing neighborhood around the winner provides an encoding of relationships, smoothing and interpolating between disparate inputs.

This interpolation is leveraged in this system to locate feature areas at a given "SOM distance" from any specified input. In operation this means the system (either the user or a process in the "decision" module) can specify a relationship distance and the SOM-based memory will return one or many feature vectors that specified distance away from the most recent search vector (typically the most recent input vector). The relationship distance is used as the magnitude of a vector projected onto the SOM, originating at the winning node of the SOM search. The orientation of the vector is currently seeded randomly, although this is seen as a weak point of the design. The node indicated by the termination of the vector on the SOM surface is sent to the decision module.

This "relationship distance" search allows the system to control the degree of relatedness between output and input in a continuous fashion. The degrees are relative within feature space, and susceptible to idiosyncrasies of the SOM model, but are arguably consistent and transposable across the SOM space. That is, a magnitude of 0 (for the relationship distance) will return the closest match to the input, while a 1 will be slightly distant, and a 5 will be much more distantly related. Additionally, this will have a similar result for any search input, anywhere on the SOM.

3.2.3 Decision

The Decision module evaluates the results of the memory bank look-ups and controls the playback module. Fundamentally, the decision module is moving closer and further away from mimicking, or playing in unison with the human partner.

This decision process is informed directly by the amount of learning or adaptation the SOM is undergoing during performance. As previously noted, with every input the SOM finds the best match and uses that input to retrain parts of the network, reducing the distance (in feature space) between the winning node, its neighbours and the new input. The difference between the retrained state and the previous state is summed and used as a measure of "learning."

Based on theories of intrinsic motivation [16, 17], this learning value over time is considered to be analogous to the Kolmogorov complexity [18], or information density of the input stream.

Control in the decision module is affected by setting a target "learning rate" which the system is trying to maintain. If input is sufficiently complex to produce a higher learning rate the system will steer towards unison with the human, allowing the musician to guide the musical movement. However, if the input stream is predictable by the system (i.e. not enough learning occurs), the decision module increases its divergence from the input, seeking more complexity and to produce novel musical movement. This later case is particularly notable when the musician is silent, causing the system to gradually explore further and further until the human provides new, stimulating input.

3.2.4 Playback

The system's output module comprises a 4-voiced synthesis engine employing audio file playback. In this way all of the sounds the system can produce are a comprehensive set of all sounds the system has heard. The playback voices are guided by the decision module, which provides a target feature vector that the playback module attempts to match. This is accomplished by searching in the k-d tree for the closest known match to the proposed feature vector. The corresponding location in the audio database is loaded and used for playback. The four voices are employed to ensure clean crossfading between different files and when changes would otherwise produce undesirable audio artifacts.

4. DEVELOPMENT

The system was implemented in Max, due to the availability of the ml.* library. System requirements were produced and evolved over many sessions, as the potentials of the components became clear. An original implementation of the k-d-tree sort and search was developed as a Max external for this project, and it is being released freely for creative use along with this text.

Development took place in an iterative process with the authors using musical material as a research instrument into the possibilities and capabilities of the system.

5. FUTURE WORK

Future *Flights* efforts with ML will include expanding on the ability of the system to create sonic variety by encoding other musical features, such as rhythm. Given the openness of the score as it relates to instrument/voice selection, many different versions of the work could create a large range of sonic material suitable for manipulation in a variety of performance scenarios.

When implementing the variations, aesthetic considerations stem from the arrangement of performed material by artists and the ML system. Audiences classify different musical features unequally, so the more abstract implementations of the effect can present difficulties when conveying the intended meaning. This will remain a problematic factor unless a clear method of articulating the ML voice, or sound is established in a performance context. More performance research and rehearsal is needed to establish best practices for this type of aesthetic communication.

Additionally, over time, the methodology employed in this work implies that the assembly/rehearsal process will also draw inspiration from the mechanics of the ML system. The performance instructions could then be adapted/expanded to explore additional musical ideas allowed by the capabilities of the effect.

6. CONCLUSIONS

ML algorithms provide a way to add variation to static deterministic systems, increasing computational approaches to creative expression. Challenges encountered while following these approaches include the establish-

ment of logical connections between the output of the system and the musical context, and the controllability of the system. Further efforts in broadening the pallet of options for the ML system, combined with more rehearsal/exploratory hours with live musicians performing the variations will yield a broader range of musical possibilities. Solutions to these issues shape the aesthetic outcome for the whole performance, as one issue affects the perceived integration of the effect by the audience, while the other affects the interaction between system and performer.

7. REFERENCES

[1] Cope, David. A Musical learning Algorithm. *Computer Music Journal. 28, no. 3* (Autumn, 2004), 12-27.

[2] Raphael, Christopher. A probabilistic expert system for automatic musical Accompaniment, *Journal of Computational and Graphical Statistics*, 10, no. 3 (September 2001), 487-512.

[3] Melo, Andres; Drever, John; Wiggens, Geraint. Electroacoustic performance interfaces that learn from their users. *Academia.edu*, http://www.academia.edu/2813098/electroacousticperformanceinterfacesthatlearnfromtheiruser, 2005.

[4] Lewis, George. Too many notes: computers, complexity and culture in voyager, *Leonardo Music Journal*, 10 (2000), 33-39.

[5] Weinberg, Gil; Driscoll, Scott; Toward robotic musicianship. *Computer Music Journal*, 30 no.4 (Winter 2006), 28-45.

[6] Van Nort, Doug, Pauline Oliveros, and Jonas Braasch. "Electro/Acoustic Improvisation and Deeply Listening Machines." *Journal of New Music Research* 42, no. 4 (2013): 303–24.

[7] Deal, Scott; Sanchez, Javier. Integration of machine learning algorithms in the computer-acoustic composition Goldstream Variations. *Proceedings, of the International Computer Music Conference. (*2013), 254-257.

[8] Fiebrink, Rebecca, Perry R Cook, and Dan Trueman. "Play-Along Mapping of Musical Controllers." In *Proceedings of the International Computer Music Conference*, (2009).

[9] Smith, Ben. Unsupervised Play: Machine Learning Toolkit for Max. In *Proceedings, New Interfaces for Musical Expression*, (2012).

[10] Smith, Ben. "The Self-Supervising Machine." In *Proceedings of New Interfaces for Musical Expression*, (2011).

[11] Shepard, Roger N. "Circularity in Judgments of Relative Pitch." *The Journal of the Acoustical Society of America* 36, no. 12 (2005): 2346–53.

[12] Jehan, T. Tristan Jehan's Max/MSP Stuff. http://web.media.mit.edu/~tristan/maxmsp.html

[13] Bentley, Jon L. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (September 1975), 509-517.

[14] Kohonen, T. "The Self-organizing Map." *Proceedings of the IEEE* 78, no. 9 (1990): 1464–80.

[15] Liu, Y., R.H. Weisberg, and C.N.K. Mooers. "Performance Evaluation of the Self-organizing Map for Feature Extraction." *Journal of Geophysical Research-Oceans* 111, no. C5 (2006).

[16] Schmidhuber, Jürgen. "Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes." *Anticipatory Behavior in Adaptive Learning Systems*, 2009.

[17] Smith, B.D., and G.E. Garnett. "Reinforcement Learning and the Creative, Automated Music Improviser." In *Proceedings, EvoMUSART*, (2012).

[18] Kolmogorov, Andrey (1963). "On Tables of Random Numbers". *Sankhyā Ser. A.* **25**: 369–375.