

Unsupervised Play: Machine Learning Toolkit for Max

Benjamin D. Smith
University of Illinois at Urbana-Champaign
Institute for Advanced Computing Applications
and Technology
Urbana, Illinois
bdsmith3@illinois.edu

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

ABSTRACT

Machine learning models are useful and attractive tools for the interactive computer musician, enabling a breadth of interfaces and instruments. With current consumer hardware it becomes possible to run advanced machine learning algorithms in demanding performance situations, yet expertise remains a prominent entry barrier for most would-be users. Currently available implementations predominantly employ supervised machine learning techniques, while the adaptive, self-organizing capabilities of unsupervised models are not generally available. We present a free, new toolbox of unsupervised machine learning algorithms implemented in Max 5 to support real-time interactive music and video, aimed at the non-expert computer artist.

Keywords

NIME, unsupervised machine learning, adaptive resonance theory, self-organizing maps, Max 5

1. INTRODUCTION

While ML applications are providing significant developments and advances in interactive music performance, the tools continue to demand high degrees of expertise. Typical applications today are either crafted by a sole technician/artist [10, 12] or by a team of developers and artistic creators [9]. However prepackaged libraries and toolboxes for popular development environments are becoming increasingly prevalent [3, 4], setting the stage for wider adoption of these techniques within the larger community.

The ML techniques available to the interactive artist today consist predominantly of supervised models and algorithms. This approach allows the user to carefully cull training data in order to achieve high degrees of accuracy and repeatability in the system's operation. Further, common limitations, such as extremely long training periods and high computational requirements, are being overcome in recent implementations [3], enabling live, interactive learning and model construction.

Adaptive or unsupervised techniques, which are largely absent in available packages, can provide a number of advantages in certain situations [8]. Unlike supervised techniques, the adaptive models learn incrementally, and training is always additive, avoiding the requirement of repro-

cessing the entire training set at each iteration or *epoch*. They also learn immediately, while supervised techniques often require thousands of epochs before they converge to a suitable tolerance. Unsupervised algorithms arguably map well to models of human perception and are self-organizing, being able to function without any external intervention.

Towards making the breadth of ML models available to the interactive musician and artist we developed and released a small library of unsupervised techniques for the Max environment¹. Self-organizing maps (SOM)[6], adaptive resonance theory (ART)[1], multi-layer perceptron networks (MLP), and spatial encoding techniques [2] are all available as pre-compiled binaries and/or java objects for Max. The library is freely distributed for non-commercial applications. We present a discussion of the affordances and limitations of these models as they relate to interactive music to serve as a brief tutorial for the non-expert user on the use of unsupervised ML techniques for real-time performance.

2. ML.SOM

The self-organizing map (SOM) [6] 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. Visualizing the map can lead to quick, intuitive insights into the organization of the source data, revealing clusters of importance and interest. 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. The SOM may also be considered a nonlinear generalization of principle component analysis over which the SOM arguably provides many advantages [7].

When an input is presented to the SOM a search is performed to locate the most similar (i.e. closest, using some conventional distance measure) or winning node in the map. Learning is then performed, adapting the winning node and its neighbors to more appropriately represent this new input. The learning is calculated as a gradual reduction of the distance between the input and the matching map node (the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NIME'12, May 21-23 2012, Ann Arbor, Michigan.

Copyright remains with the author(s).

¹<http://cycling74.com>

process can be understood as a simple low-pass filter), and this rate of adaptation is controlled by the map's *learning rate*. The signature topographical appearance of the SOM is a result of adapting nodes in a gradually decreasing neighborhood around the winner. This effect is controlled by a *neighborhood radius* parameter and the adaptation moves nodes adjoining the winning node with proportion decreasing towards the edge of the neighborhood (we employ a linear gradation). The SOM is typically initialized with random values, however this can produce markedly divergent classifications (also dependent on input analysis order). Other alternatives are to initialize the map uniformly or use some form of prediction (such as principle component analysis or an older SOM on a previous, exemplary data set).

Finally, the SOM may consider the age of the map and gradually become more resistant to change, eventually settling on a permanent representation of the data. Both the advantage and disadvantage of this is that data up to a certain point is retained as the basis for the mapping, remaining immune to (i.e. ignoring) divergently new data (controlled with the *solidification* property of the **ml.som**).

The results of SOM clustering can be seen in fig. 1, where the map trains on a selection of colors. One employs three colors (cyan, magenta, and yellow) which settle at three extremities of the map, overlapping to produce the range of fully saturated colors. The other uses eight colors (the initial three plus red, green, blue, white, and black). Black and white are pushed to the corners (the selection of the top corners is coincident with the random seeding of the map) and the remaining six colors organize themselves in a classic color-wheel pattern. The SOM is highly dependent on initialization state and the input data order and the significance of the resulting perceived relationships in the map may vary accordingly. Randomizing and retraining the SOM produces different orientations of the colors (and different corners for black and white), but the dominant patterning remains consistent.

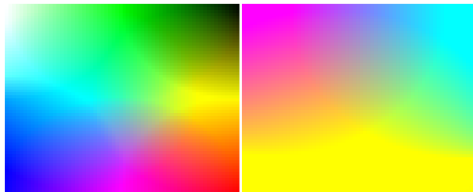


Figure 1: SOM with 64x64 nodes trained on eight colors (left) and three colors (right).

2.1 Example

As an example application we describe a minimal system to cluster timbre exemplars from a live audio stream (see fig. 2). The objective is to enable a live performer to play a range of material and have the SOM bring samples of similar sounds (in timbre) together on the map. This is a useful example because timbre similarity measures remain an open problem in music perception theory. **ml.som** aims to provide an easy-to-use, robust interface, and lists received by the object are treated as new inputs resulting in immediate matching and learning.

We start with the assumption that digital audio is being processed in Max and that it has been analyzed for salient features (spectral centroid, loudness, noisiness, Bark bands, etc.). These features, ensuring that each value is scaled to (0-1), are then fed directly to the SOM. We configure the SOM with the following arguments:

- 64 nodes in width and height,

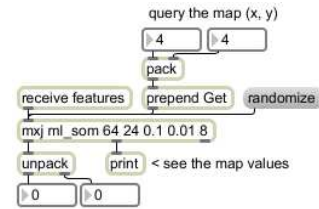


Figure 2: Minimal SOM patch to classify input features (using java **ml.som object).**

- 24 items in the input, feature vector,
- learning rate of 0.1 (or 10%),
- solidification rate of 0.01 (or 1%),
- neighborhood radius of 8 nodes.

In fig. 2 the *randomize* message will initialize the map to a random state and should be used before operation commences. Once playing is underway the 2D locations that are being trained will be displayed in the number boxes descending from the left-most outlet. We leave it to the end user to define a function for the use of this data (perhaps: record the coordinates at certain points during the performance, calculate a distance measure, reflecting a degree of timbral similarity or difference, and using this result to drive a synthesis engine).

The right outlet of the **ml.som** (middle outlet of the java object) produces the modified, trained state of the matching node (based on the most recent input or *Get* message). This enables algorithmic analysis and/or display of the map's data.

It is expected that the playing of the live musician will effectively trace shapes or patterns on the surface of the map, traversing figures that map reliably to given musical statements (such as articulations, register, and dynamic changes). These could easily be rendered visually (using an *lcd* object, for example), to give the user an idea of the nature of the mapping. Once a regularity is observed it could be connected to another layer of machine learning (using another **ml.som** or an **ml.art**) to learn these higher level patterns and map to analogous functions in the performance system.

3. ML.ART

Adaptive Resonance Theory [1] algorithms were initially proposed as a computational model of human attention, employing a neural network. Like the SOM, ART compares new input feature vectors with known category nodes and adaptively trains the network. However, ART nodes do not influence one another during training (they are not connected in the sense that the SOM employs), and there is no dominant spatial metaphor at work. While the SOM maps feature space in a continuous fashion across the map (i.e. intermediary points between nodes in the SOM network could be interpolated) the ART encodes a continuous area of feature space into each node (**ml.art** implements the fuzzy ART version).

When a winning category node is identified during input presentation the node must pass a *vigilance* test before learning can commence. The vigilance process verifies the area that the winning category would represent if training were to proceed and ensures that the category does not grow beyond a preset limit (set with the *vigilance* parameter). If the category will remain within the limit it is allowed to adapt and learn the new input, much in the same fashion as the SOM (using a *learning rate* parameter). However, if

the category would expand to encompass too much of the feature space then the node is removed from consideration and the search for a winning node is performed again.

Thus the ART is ideal for locating categories within continuous data streams and the granularity of the analysis can be easily adjusted. As with the SOM the selection of features is important but in the ART the full dimensionality of the category relationships is preserved, allowing further analysis of the ART results at any point (see [11], for example).

The ART is computationally efficient and, because new nodes may be added as needed, capable of classifying extensive data sets with a wide range of specificities (limited only by hardware). It is arguably a suitable model of human long-term memory processes [5, 10] and thus is useful in computational models of perception. While the node index of the ART has only internal significance it is trivial to obtain the resonance (i.e. distance measure) of each category with each input. This resonance, or fit, can describe many useful aspects of the input, such as how clear the fit is (difference between the peak and the average resonance), how significant the input is (sum of all resonances), and how this fit compares to previous fits (indicating movement in classification space). This last measure can potentially give a strong sense of sectional movement in a piece, indicating transitional passages or sudden changes.

While the ART is not susceptible to initialization divergences (all untrained nodes are considered null), it is highly dependent on the input data order. The classifications, and especially the boundaries defined by the ART nodes, can be dramatically different if the inputs are presented in a different sequence. This can be both a strength, in the case of music analysis where the identity of a piece or improvisation demands a certain sequence, and a weakness, appearing to be inconsistent where supervised models are consistent.

ML.art has one inlet and two outlets and accepts lists which are treated as input feature vectors. Here the input feature vector size does not need to be specified in advance as the nodes are only populated as data is received and the network will continue to grow as needed to encode all of the received data. By default the ART will always match the best sub-set category to the received input, although this behavior can be defeated by setting the *choice* parameter > 0 , where higher values enforce more restrictive matches. All of the parameters can be set dynamically with appropriate messages and the **ml.art** can be cleared at any point. Upon successful learning of a new node **ml.art** outputs the winning category ID as well as the resonances of all the trained categories.

3.1 Example

We now describe the construction of a small system to analyze, learn, and identify melodic pitch sequences (modeled on [5, 10], see fig. 3). Apart from **ml.art**, this will require a method for producing a sequence of pitch data. Once a pitch sequence is available it is reduced to pitch classes (taking pitch modulo % 12).

In order to perform pattern extraction and matching on this pitch sequence the ART needs a uniformly dimensioned representation that captures both the pitch classes and the ordering of the notes. This is accomplished using **ml.spatial**, described below. For now, we place an **ml.spatial** object at the bottom of the short processing chain (the resulting feature vector for a chromatically descending melody is displayed in the chart on the middle-right of fig. 3).

Now, we add an **ml.art** object with the parameters “0 0.5 0.85” (corresponding to *choice*, *learn rate*, and *vigilance*, respectively). These are set to encourage deep set-subset

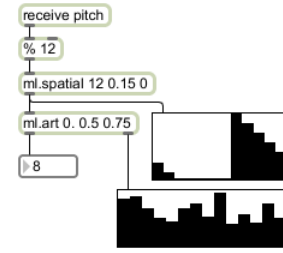


Figure 3: Max 5 patch employing **ml.art** and **ml.spatial** to identify melodic patterns.

searching, a mid-rate learning speed, and fairly tight category generation, respectively. The **ml.spatial** connects directly to the **ml.art**.

Finally, the category index for any given input is displayed in the number box at the lower left. Playing into the system now causes the ART to extract and learn melodic patterns. For example, a musician could choose several short melodic fragments (5-7 notes in length) and make a note of the category index numbers produced for each. During an improvised performance the occurrence of these indices could then be used to trigger desired actions.

This event oriented interaction model can be further augmented to incorporate continuous variation by measuring the relative resonance of each target node for any given input. The right outlet of **ml.art** produces the resonance vector resulting from each input as it is presented to the ART (shown graphically in the lower-right chart in fig. 3). Unpacking this list and watching the value at the position correlating with the target indices will give a measure of how closely the current input matches, or resonates with, the trained pattern categories (in fig. 3, for example, the most recent input matches category 8, shown in the number box, resonates strongly with categories 1 and 2, but much less so with categories 4, 5, 7, and 9). Thus a desired melody or pitch sequence, once learned by the ART, can be used as a reference during a performance continually answering “how close is the current material to what I have already learned?”

4. ML.SPATIAL

As has been mentioned, the identification, selection and encoding of features is a key component of ML techniques. Many desirable traits can simply be scaled to an appropriate range and fed directly into one of the models described above, however this is not always the case. For example, simply treating pitch as a continuous frequency value misses the functional relationships inherent in tonal musics. The technique of spatial encoding provides a ready solution, allowing the ML model to learn the relationships between elements in a time ordered sequence.

Spatial encoding arose in natural language analysis where letters of words are encoded into vectors and used as patterns for classification [2]. The same principle can be applied to music, using any tokens from a set (such as scale degrees or pitch classes) [5]. The encoding is accomplished using a single layer neural net model with an attenuation feed-back component. Each token in the given data set is assigned to a node in the network (i.e. twelve nodes would be used to encode pitch classes). When a token is presented to the network the associated node is fully energized (value set to 1) and a suppressant is applied to all of the other nodes, modeling the attention of the network focusing on the newest input (see fig. 4). This is typically accomplished

by reducing the energy of each node by a small amount (in a linear fashion, say by 0.15), or damping the network as a whole (exponentially, multiplying each node by, say, 0.85). The product of the encoding is a uniformly dimensioned vector-representation of the ordering of the most recent tokens (typically 7 ± 2 when modeling typical human short-term memory).

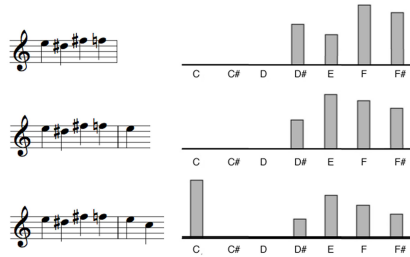


Figure 4: Spatial encoding of short melodic fragments.

Once the encoding is complete the vector can be processed by a ML model. A spatially encoded token string makes set-to-subset comparisons trivial and makes identification of reordered sets possible as well (retrograde, extension, contraction, and ornamentation thus become transparent to the classifier). Each of these will result in encoded vectors that have some similar elements, enabling ready identification by an SOM or ART, for example.

However, ambiguities can occur, for example when a token appears in a string multiple times and earlier appearances are overwritten by the later ones. The effect of that earlier token is still present in the decay of the other nodes, yet depending on the required precision of the system this can have a detrimental effect.

A dynamic activation mode [11], incorporated in **ml.spatial**, attempts to reduce this negative aspect. This mode activates a node an amount proportional to the calculated attention merit for the presented token (the merit value must be input along with the token). This method obscures the ordering of the tokens yet it allows a prominent token to receive a large amount of attention while insignificant tokens are only minimally represented (this mode is activated with an appropriate message to the **ml.spatial** object).

The parameters for controlling **ml.spatial** are: the number of tokens in the input dataset (26 for english text, 12 for pitch classes, etc.), the decay rate (strength of attenuation), and the choice of a linear versus exponential decay model. The decay rate controls the length of the network's memory and can be calculated as one over the desired length (to retain 7 tokens the decay rate should be set to $1/7$ or 0.143). The linear model subtracts this amount from each node at each time step (new input presentation) while the exponential model multiplies each node by one less the decay rate.

Operation in Max is simply a matter of inputting an integer representing the token under consideration and receiving the output feature vector (as a list of floats). This can then be directly routed to **ml.som** or **ml.art** objects.

Proven cases for spatial encoding include melodic scale degrees, tonal intervals, pitch classes, and interval classes [5, 10, 11]. Timbre space and rhythmic modeling may be ready areas for encoding as well. Additionally, it becomes possible to route the output of one ART (the winning category ID) into a spatial encoder and create a multi-layered ART system [5, 12], which serves to track hierarchical structure within the input.

5. CONCLUSION

Currently none of the **ml.x** objects explicitly support state preservation between sessions. We plan to implement model export capabilities to enable rapid saving and loading of pre-trained states. Additionally, we intend to extend the library to include more techniques as they are identified or requested and a PD release of this library is also under consideration.

We have described the functionality and theory behind several powerful ML techniques implemented in a new toolbox for Max. Simple yet pertinent examples were described with the goal of providing access to these tools for non-expert users, through the form of quickly and easily implemented programs. Ultimately we hope that this work will contribute to, and further enable the continued exploration of the new aesthetic possibilities afforded by ML techniques in interactive computer music.

6. ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of eDream and the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

7. REFERENCES

- [1] 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.
- [2] 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.
- [3] R. Fiebrink, P. R. Cook, and D. Trueman. Play-along mapping of musical controllers. In *Proceedings of the International Computer Music Conference*, 2009.
- [4] N. Gillian, R. Knapp, and S. O'Modhrain. A machine learning toolbox for musician computer interaction. *Proceedings of the 2011 International Conference on New Interfaces for Musical Expression (NIME11)*, 2011.
- [5] R. O. Gjerdingen. Categorization of musical patterns by self-organizing neuronlike networks. *Musical Perception*, 1990.
- [6] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [7] Y. Liu, R. Weisberg, and C. Mooers. Performance evaluation of the self-organizing map for feature extraction. *Journal of Geophysical Research-Oceans*, 111(C5), 2006.
- [8] M. P. A. Page. Modelling the perception of musical sequences with self-organizing neural networks. *Connection Science*, 6(2 & 3):223–246, 1994.
- [9] M. Schedel and R. Fiebrink. A demonstration of bow articulation recognition with wekinator and k-bow. In *Proc. International Computer Music Conference*, 2011.
- [10] B. Smith and G. Garnett. The self-supervising machine. In *Proc. of New Interfaces for Musical Expression*, 2011.
- [11] B. Smith and G. Garnett. Machine listening: Acoustic interface with art. In *Proc. of SIGCHI Intelligent User Interfaces*, 2012.
- [12] B. Smith and G. Garnett. Reinforcement learning and the creative, automated music improviser. In *Proc. of EVOMUSART*, 2012.