Wekinating 000000Swan: Using Machine Learning to Create and Control Complex Artistic Systems

Margaret Schedel
Stony Brook University
Stony Brook, NY
margaret.schedel@stonybrook.edu

Phoenix Perry
NYU Poly
New York, NY
phoenix@areyoudevoted.com

Rebecca Fiebrink
Princeton University
Princeton, NJ
fiebrink@princeton.edu

ABSTRACT
In this paper we discuss how the band 000000Swan uses machine learning to parse complex sensor data and create intricate artistic systems for live performance. Using the Wekinator software for interactive machine learning, we have created discrete and continuous models for controlling audio and visual environments using human gestures sensed by a commercially-available sensor bow and the Microsoft Kinect. In particular, we have employed machine learning to quickly and easily prototype complex relationships between performer gesture and performative outcome.

Keywords
Wekinator, K-Bow, Machine Learning, Interactive, Multimedia, Kinect, Motion-Tracking, Bow Articulation, Animation

1. INTRODUCTION
Obsessed with electronics, rare birds, myth, Native American art, pagan ritual, fetish, punk, and tribal percussion, 000000Swan is an experiment in performing process and interaction. We create high-impact, hard-to-predict events beyond the realm of normal expectations, performing on a variety of electronic instruments including keyboards, a JazzMutant Lemur, a Zeta cello with a sensor bow, and a Kinect. We are able to quickly create interactive audio and visuals by harnessing the power of machine learning with Wekinator. In this paper, we discuss how we created the interactive audio and visual elements for the song *Monster*.

2. HARDWARE AND SOFTWARE

2.1 Wekinator
The Wekinator [2][3] is a freely available software environment designed to facilitate the interactive application of supervised learning to real-time problem domains, including music.[1] Supervised learning algorithms are a family of machine learning algorithms capable of using a training dataset to produce a model (see, e.g., [1]). This model can be understood as a function capable of producing some output value (e.g., a gesture label, such as "staccato") from some input value (e.g., a feature vector computed from sensor bow outputs). The training set consists of a set of example input-output pairs (e.g., each pair might consist of a single feature vector and the true gesture label that should be applied to that feature vector). Supervised learning has been an effective tool for building models in many problem domains in which labeled training data is available, but where the relationship between features and labels is too complex to specify explicitly in code. Musical gesture identification and mapping creation are two such domains in which prior work has found supervised learning to be useful (e.g., [6][9][12]).

The Wekinator provides a graphical user interface for collecting and editing training data, training learning algorithms, and running trained models to produce outputs from inputs in real-time. Users create training examples by specifying the target output (e.g., gesture class) in the GUI and demonstrating the corresponding gesture or other input signal; features are extracted from the user's input and saved with the target value. Wekinator includes implementations of standard discrete classification algorithms (k-nearest neighbor, decision trees, support vector machines, and AdaBoost.M1), as well as multilayer perceptron neural networks for regression. Users are able to interactively change algorithms, algorithm parameters, and selected features. Significantly, users are also able to influence model behaviors by adding, deleting, and editing the training examples. Compared to other machine learning tools, the Wekinator was designed to more explicitly support rapid, iterative model design through interactive changes to the training dataset [3].

Once a user has created a model by training a chosen algorithm, (s)he can run the model to produce predicted outputs for incoming feature vectors that are extracted in real-time. In our bow gesture classification system, for example, the user can execute different types of bow gestures using the K-Bow and observe the model's predicted output over time.

2.2 Kinect
The Kinect is a hands-free accessory for Microsoft's Xbox 360. It uses an RGB camera in combination with a depth sensor and multi-array microphone to enable users to interact with video games without a physical controller.[2] It was released in the USA in November of 2010 and was quickly hacked to enable units to send data directly to computers via the USB port. In our performance, we use the depth data as input to supervised learning models created by the Wekinator, allowing us to use body movement to control and trigger both audio and video.

2.3 K-Bow
The K-Bow is the first commercially-developed, mass-produced sensor bow for string players [7]. It contains 1) a three-axis accelerometer located inside the frog, which senses tilt and acceleration of the bow in space; 2) a grip sensor that perceives changes in the grip pressure and surface area of the

[1] http://wekinator.cs.princeton.edu/

[2] http://www.xbox.com/en-US/kinect

cellist's bow hand; 3) an angle-sensitive pressure sensor located at the junction between the bow hair and the frog, which measures changes in the tension of the bow hair; and 4) an infrared detector inside the frog, which measures the bow position and angle relative to a circuit board and IR emitter mounted under the fingerboard.

The K-Bow ships with a software suite, K-Apps, which receives sensor values from the bow. This software provides a GUI interface for sensor calibration and sends sensor values to other software programs via OSC or MIDI. We use data from the K-Apps as input to several Wekinator models to control and trigger audio and visuals in our performance.

2.4 Audio and Visual Software

Ableton Live is a Digital Audio Workstation optimized for live performance.[3] Using data from the keyboards, Lemur, Zeta Cello, and Wekinator we are able to control audio processing, launch samples and loops, as well as play software synthesizers while simultaneously controlling synthesis parameters. For example, the lead singer might be playing keyboards while data from the K-Bow adjusts the distortion on the patch she is playing.

Unity is an integrated graphical environment for creating 3D games and animations.[4] Its game engine runs on multiple platforms including Windows and OS X, a web plug-in, iDevices, and most commercial game consoles. Using data from the Wekinator we are able to control an interactive game environment, launching visuals, changing colors and camera angles, and creating generative graphics such as particle systems to create visuals for *Monster*.

2.5 Data Flow

Our data flow is illustrated in Figure 1. K-Apps receives K-Bow sensor outputs and forwards them to a standalone feature extractor, which extracts features (e.g., minima and maxima, first- and second-order difference) and sends them to Wekinator via OSC [11]. Simultaneously, rudimentary features are extracted from the Kinect to roughly describe the 3D location of the human performers, and these are sent to the Wekinator as well. Certain Wekinator models are trained to create and control Ableton Live sounds in response to features extracted from the K-Bow and/or Kinect, and other Wekinator models are trained to drive aspects of the Unity game engine.

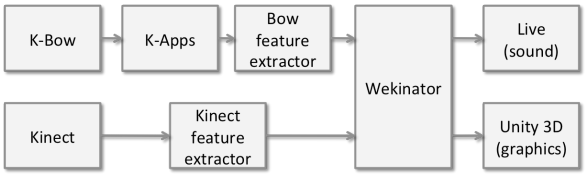

Figure 1. Data Flow for 000000Swan

3. DISCRETE CLASSIFICATION OF BOW ARTICULATIONS

3.1 Prior Work using K-Bow and Wekinator

Prior research has shown the discrimination of string bow strokes and articulations to be tractable using sensor bows and machine learning (e.g., [9][12]), though this work has not studied the production of classifiers that were later used in live performance. In our own prior work, we used the Wekinator to create eight bow stroke classifiers for the 000000Swan cellist using the K-Bow. For example, our articulation model classifies seven standard bow articulations (see [4]): legato (smooth and connected), marcato (onsets emphasized and slightly detached), spiccato (enunciated and percussive), riccocet (bouncing, rapid notes), battuto (struck with the wood of the bow), hooked (re-articulation of notes without a change in bow direction), and tremolo (rapid alternation of up-bows and down-bows). The classifiers were constructed to identify articulations played on any string of the cello and to be reasonably robust to changes in horizontal and vertical bow position (i.e., frog, middle, tip; sul tasto, sul ponticello), bow pressure, and bow speed. The articulation classifier was the most complex model that we built, and it achieved a 98.8% cross-validation accuracy and a subjective quality rating by the cellist of "9" out of "10."

3.2 Discrete Classification Experience with 000000Swan and Wekinator

In performance we found we needed a way for the cellist to trigger discrete events, much like a button on the Lemur. During difficult vocal passages for the lead singer, we decided it was much more important to focus on the vocal line, versus attempting to both sing and trigger, therefore the cellist needed to be able to trigger samples. We tried using particular notes on the cello, but the unique notes for triggering stood out from the rest of the cello line. Triggering from bow position did not give satisfactory results; the only way to make it consistent left only two possible triggers at the very tip and directly at the frog. By using the seven identifiable articulations from the Wekinator in combination with string information and extreme bow position (i.e. frog, tip, ponticello, and sul tasto), we were able to recognize 112 unique triggers which we use for multiple songs in a set.

3.3 Discrete Classification in *Monster*

One way we use the bow articulation triggers in *Monster* is to change the color of the visualization. The bass line is consistent throughout each verse, but the first verse uses a legato bowing to produce a purple visualization, and the second uses marcato to create white particles.

We also use bow articulations to trigger samples; almost inaudible riccocets, tremolos, and batuttos on the A and D string in the ponticello and sul tasto positions enable the cellist to trigger 12 discrete audio samples varying in length from 0.2 seconds to a minute-long ambient sweep towards the end of the piece. This ability to add elements during the performance is important to us; an integral element of the 000000Swan aesthetic is to make each live show unique.

4. CONTINUOUS CONTROL USING KINECT

4.1 Prior Work in Continuous Control with Wekinator

The Wekinator has previously been used by other composers to create interactive systems in which performers' gestures continuously control sound synthesis parameters [8][10]. In those compositions, as in components of our own work, composers used the Wekinator to prototype, refine, and perform with many-to-many mapping functions built from neural networks. Unlike prior compositions, we have combined continuous and discrete control mechanisms, and we engage gestures of multiple performers to control both audio and visuals.

[3] http://www.ableton.com/
[4] http://unity3d.com/

4.2 Continuous Control Experience with 000000Swan and Wekinator

Our lead singer has a dance background, and we often work with aerialists. We wanted a way to use body movement to control aspects of the performance. We programmed several tracking systems in Max/MSP/Jitter/SoftVNS, but we were unhappy with the results. Either the mapping from gesture was too direct and uninteresting, or else the tracking was not robust. In addition, the system ran very slowly. Using the Wekinator circumvented these problems. Since the Kinect sends formatted vision tracking data directly into Wekinator the environment is very responsive. Unity runs directly on the GPU so we are able to create complex visuals without taking up too much of the CPU, leaving more power for Ableton.

With Wekinator we are able to quickly train models to drive sound and visuals in response to gestures performed in front of the Kinect. We can train models for specific spaces by creating training examples in the venue before the performance. For example, we may use the downward motion of the aerial dancer to manipulate the EQ on a synthesizer patch in Live. The range of the dancer's height changes depending on the elevation of the rigging. We only need to give Wekinator two training examples—one at the top of the dancer's range, mapped to a narrow EQ of 2, and one at the bottom, mapped to a wide EQ of 18—to recalibrate the height-EQ model for a new venue. This is a simple mapping, but we also use the Wekinator for many-to-many mappings as discussed in the next section.

We also use the Kinect to track the musicians' movement to influence sound and visuals. Previous tracking systems were very dependent on costumes and lighting; using the depth sensor from the Kinect, we have eliminated lighting as a variable. Since the Wekinator is so easy to train we can create models in multiple costumes, making our performances more robust.

4.3 Continuous Control in *Monster*

In *Monster*, we use a particle generator to create interactive visuals. One layer on top of the particle generator is a spiral of triangle shapes. The position of the triangles is controlled by the position of the right arm of the lead singer. The Kinect is able to track this variable through the entire field of the camera. Using neural networks to create continuous mappings from arm position to triangle position allows the visuals to respond dynamically to gradual changes in the singer's movement.

We also use body position to control camera parameters in Unity. Using a set of five Wekinator models, we are able to create a many-to-many mapping between performer gesture and Unity's camera focus, angle, and 3D position. The same position features are used to drive seven of Live's processing parameters. Performers thus affect the visuals and audio in complex and dynamic ways that which would be difficult, if not impossible to code by hand.

We train these models in the venue using four types of training examples: 0) Standing close, cello playing arco, left hand high on the strings 1) standing close, cello playing pizzicato, left hand low on the strings 2) standing far apart, cello playing arco 3) lead singer crouching, cellist leaning backward 4) lead singer with arms in the air, cellist kneeling. We know basically what visuals and audio processing will result from these position states, but we do not know how the "in between" states will react. We know generally what will happen, but sometimes the results surprise us. For example, if the lead singer is crouching and the cellist is kneeling, the visual state may be somewhere between (3) and (4), but we don't know for sure until we experiment with the trained models. This poses a creative challenge; we want the system to be predictable and reproducible, while remaining engaging. This type of mapping is also rewarding in that, by creating "meta-sensors" driven by the actions of both performers, each member has her own role in shaping the collective experience.

5. DISCUSSION

5.1 Advantages of Machine Learning

000000Swan is extremely pleased with the Wekinator. Previous interactive systems we developed were not robust over multiple venues and costumes, we found it difficult to program complex results, and we felt we were spending more time coding than working on the music and visuals. With the Wekinator we are able to take complex streams of information from multiple controllers and quickly program audio and visual responses. We use both concrete classifiers as triggers and continuous classifiers to transform between states.

We see five real advantages to using interactive machine learning in our multimedia performance:

1) **Efficiency in design:** We no longer have to parse complex sensor information ourselves. Instead of trying to understand eight variables coming in from the K-BOW every 10ms, and thousands of IR points coming from the Kinect every 33ms, we can think about the bigger picture and let Wekinator handle the details.

2) **Customizability:** The Wekinator is fast to train; we are no longer anxious about how our system will respond in different venues. We simply train the program in each setting, in costume. We create models in our dress rehearsal but retain the output response.

3) **Supporting complex performer-performer interactions:** The Wekinator does not distinguish between the types of data coming in; therefore, we can track multiple sensors at the same time to create "meta-sensors." This augments the interaction between the performers.

4) **Supporting complex mapping strategies:** We can create both discrete triggers and continuous control in the same program, and Wekinator's neural networks create complex, interpolating systems with many-to-many mappings without a lot of programming.

5) **Rapid prototyping:** We can re-train and experiment quickly with different models for the same sensors to control the sound and visual environment.

To some extent, the practical advantages that machine learning offers in creating customizable, complex mappings without explicit programming are inherent to the use of a generative mapping strategy (e.g., see [5]). In our experiences, these benefits are also contingent on the ability to rapidly create, explore and change the machine learning models. A less interactive system that did not allow us to experiment with changing training examples, that took a long time to train, or that made it difficult to quickly test models by running them on real-time inputs would be significantly less useful to our work.

5.2 Disadvantages of the Wekinator

Although we are happy with the Wekinator, there are some disadvantages that we have had to work around.

1) There is no explicit support for triggering. In Max/MSP/Jitter it is trivial to create a trigger. With the Wekinator you must go through a secondary router in order to create a trigger.

2) The Wekinator's OSC output messages aren't customizable in format. We therefore rely on a third-party

routing software (OSCulator) to translate them into the correct format for Ableton.

3) There isn't an easy way to "turn off" the output of selected Wekinator models. The easiest way to program Ableton for control by an external OSC process is to click on the parameter to control (e.g., volume) and move the controller (and only that controller). However, because Wekinator's models all continuously output values simultaneously, OSCulator was also used for this function.

In order to streamline our workflow, we are working with the creator of Wekinator to improve the software by allowing triggering and greater control over its OSC output behavior.

5.3 Other control strategies in *Monster*

We don't use Wekinator for all of the controller data in *Monster*. We a keyboard to send traditional MIDI in order to play synth pads, and we use a LEMUR to send OSC directly to Ableton, using sliders to control the volume of the singers, electronic sound and cello and buttons to launch the song, and to trigger samples. For one-to-one mappings, such as the horizontal bow position mapping to distortion on the synth pad we bypass the Wekinator and simply use the OSC data from K-Apps.

6. CONCLUSION

We have summarized our use of machine learning techniques in driving sound and graphics in our live interactive performance, *Monster*. Our use of these techniques builds on a large foundation of prior work that has demonstrated the feasibility of applying machine learning to gesture analysis and mapping creation. Through the use of the Wekinator software, we have been able to put these techniques into practice in our own work.

Although the use of the Wekinator software has required us to create several extra software modules for feature extraction, the most significant impact machine learning has had on our work is the reduction in the need to write code and the expansion of control possibilities available to us. As a result, more of our development and composition time has been devoted to exploration of these possibilities, and our attention has been more focused on cultivating the creative and artistic qualities of our work.

7. REFERENCES

[1] Bishop, C. M. 2007. *Pattern Recognition and Machine Learning*, 2nd ed. Springer.

[2] Fiebrink, R. 2011. *Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance.* PhD thesis, Princeton University.

[3] Fiebrink, R., Trueman, D., and Cook, P. R. 2009. "A meta-instrument for interactive, on-the-fly machine learning." In *Proc. Intl. Conf. on New Interfaces for Musical Expression (NIME).*

[4] Flesch, C. 2000. *The Art of Violin Playing: Book One*. Carl Fischer, New York, NY, USA.

[5] Hunt, A., and M. M. Wanderley. 2002. "Mapping performer parameters to synthesis engines." *Organised Sound*, 7(2): 97–108.

[6] Lee, M., A Freed, and D. Wessel. 1992. "Neural networks for simultaneous classification and parameter estimation in musical instrument control." *Adaptive and Learning Systems* 1706: 244–255.

[7] McMillen, K. A. 2008. "Stage-worthy sensor bows for stringed instruments." In *Proc. Intl. Conf. on New Interfaces for Musical Expression (NIME).*

[8] Nagai, M. 2010. *MARtLET*. http://michellenagai.com/Site/MARtLET.html

[9] Rasamimanana, N., Flety, E., and Bevilacqua, F. 2005. "Gesture analysis of violin bow strokes." In *Proceedings of Gesture Workshop 2005 (GW05)*. 145–155.

[10] Trueman, D. 2010. "Clapping Machine Music Variations." In *Proc. Intl. Computer Music Conference (ICMC).*

[11] Wright, M. and Freed, A. 1997. "Open Sound Control: A new protocol for communicating with sound synthesizers." In *Proc. Intl. Computer Music Conference (ICMC).*

[12] Young, D. 2008. "Classification of common violin bowing techniques using gesture data from a playable measurement system." In *Proc. Intl. Conf. on New Interfaces for Musical Expression (NIME).*