A DEMONSTRATION OF BOW ARTICULATION RECOGNITION WITH WEKINATOR AND K-BOW

Margaret Schedel
Stony Brook University
Music Department & cDACT
margaret.schedel@stonybrook.edu

Rebecca Fiebrink
Princeton University
Department of Computer Science (also Music)
fiebrink@princeton.edu

ABSTRACT
Using the Wekinator software tool for real-time, interactive machine learning [3] and the K-Bow commercial sensor bow [5], we have constructed a real-time cello bow articulation classification system. This system is capable of outputting articulation labels (e.g., “legato,” “marcato,” “spiccato”) in real-time as a cellist performs. These labels, which are output via Open Sound Control [9], may be used in conjunction with visualization or music tools in composition and live performance. Our work is distinguished from prior work in bow gesture recognition in that the Wekinator allows a musician user to rapidly build customized bow gesture models from scratch by demonstrating bowing gestures to form a training set; the user can also interactively refine these models through iterative changes to both the learning algorithms and dataset. In this paper, we briefly describe our work creating articulation models for our own use. In particular, we show that the Wekinator and K-Bow together allowed for the fast creation of accurate models. We then propose a hands-on demonstration of this work in which ICMC attendees can use the K-Bow together to interactively build their own gesture classifiers.

1. INTRODUCTION
Articulation is the “manner in which notes are joined one to another by the performer; specifically, the art of clear enunciation in singing and precise rhythmic accentuation in instrumental playing” [8]. There are several standard string instrument articulation techniques, each of which prescribes a particular use of the bow and bow arm before, during, and after contact with the strings; definitions and instructions for producing each articulation can be found in texts such as Flesch [4].

Prior work—including that by Peiper et al. [6], Rasamimana et al. [7], and Young [10]—has examined the problem of constructing computer systems capable of correctly labeling the articulations performed by a string player, given the outputs of sensors attached to the player and/or bow. That work has investigated both features (i.e., statistics computed on the sensor values, which capture relevant characteristics of the bow motion) and algorithms that, using these features, are capable of creating accurate bow stroke models. The outcomes of that prior work demonstrate that, with appropriately chosen features and algorithms, the problem of constructing a reasonably accurate bow gesture classifier is tractable.

Such bow stroke classifiers have several potential applications to composers and musicians. They may enable the construction of interactive systems that are controlled by or responsive to the articulations employed by a performer. For example, the articulations of a performer might change the rendering of an interactive animation or visualization, or they might control parameters of a sound synthesis algorithm. Such interactive systems can enable a string musician to leverage his years of musical training, because he can employ his expertise in varying articulation to explicitly control or influence the computer, rather than exercising control or influence through a dimension that is less practiced or less understood. The output of such classifiers could also be used in conjunction with audio or other signals in interactive music systems that track a performer’s progress through a score or look for cues from the performer. Accurate classifiers may even have a place in music pedagogy, where they could give students feedback about whether they are executing an articulation correctly or consistently.

Unfortunately, there has previously not existed a clear means for composers and musicians to take advantage of the prior research in bow gesture recognition and apply it to their work. In particular, many musical users who might like to employ bow gesture classifiers in their work are not programmers or are unfamiliar with the process of building feature extractors and training machine learning algorithms. We propose that the end-user interactive machine learning paradigm supported by the Wekinator [3], coupled with the K-Bow [5] and an appropriate feature extractor, can enable these users to build their own gesture models and employ them in various ways in their work. In the following sections, we describe the hardware and software that we have used to build a set of working bow gesture classifiers, which was done in the context of a larger project studying interactive machine learning in computer music [2]. We employ data collected through our process of building these classifiers to demonstrate the feasibility of building accurate and usable models quickly. We conclude with a description of our proposed ICMC demonstration, in which we will enable conference attendees to use the same software and hardware to experiment with building their own gesture classifiers.
2. HARDWARE AND SOFTWARE

2.1. The K-Bow
The K-Bow is the first commercially-developed, mass-produced sensor bow for string players [5]. It contains several sensors for measuring the position and motion of the bow in real-time. A three-axis accelerometer located inside the frog senses tilt and acceleration of the bow in space. A grip sensor senses changes in the grip pressure and surface area of the cellist’s bow hand. An angle-sensitive pressure sensor located the junction between the bow hair and the frog measures changes in the tension of the bow hair as the cellist plays the strings of the instrument. The player also affixes a small circuit board beneath the fingerboard of the instrument. This board creates an RF field and an infrared modulated wide field light cone, whose interactions with the loop antennas inside the bow stick and with the infrared detector inside the frog allow the measurement of the bow position and angle relative to the instrument. These sensors are summarized in Table 1.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x, y, z</td>
<td>Bow acceleration and tilt, measured by internal 3-axis accelerometer</td>
</tr>
<tr>
<td>Hair</td>
<td>Bow hair tension</td>
</tr>
<tr>
<td>Grip</td>
<td>Grip pressure and surface area</td>
</tr>
<tr>
<td>Length</td>
<td>Horizontal distance between frog and tip</td>
</tr>
<tr>
<td>Bridge</td>
<td>Vertical distance between fingerboard &amp; bridge</td>
</tr>
<tr>
<td>Tilt</td>
<td>Tilt of bow relative to instrument</td>
</tr>
</tbody>
</table>

Table 1. K-Bow Sensors

2.2. The Wekinator
The Wekinator is a freely available software environment\(^1\) designed to facilitate the interactive application of supervised learning to real-time problem domains, including music. Supervised learning algorithms are a family of machine learning algorithms capable of using a training dataset to produce a model. This model is essentially 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 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 can be an effective tool for building models in problem domains in which labeled training data is available, but the relationship between features and labels is too complex to specify explicitly in code. Musical gesture identification is one such domain, and the prior work in articulation classification described above has all used supervised learning to create the classifier models. A more thorough discussion of supervised learning can be found in Bishop [1].

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 label (or 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 label. Users are able to edit the training data by adding or deleting examples, as well as editing the feature or label values. Users may select among a set of standard classification algorithms (k-nearest neighbor, decision trees, support vector machines, and AdaBoost.M1) and edit the parameters employed by these algorithms. Once a user has created a model by training a chosen algorithm, she can run the model to output predicted labels 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. Figure 1 shows one Max/MSP visualization that we have used to graphically display the model’s output; here, instead of just displaying the single best label, the visualization shows the posterior probability distribution over all labels, indicating the model’s “certainty.” The Wekinator allows the user to improve models by modifying the training dataset, learning algorithm, algorithm parameters, and/or selected features.

The Wekinator is capable of receiving feature vectors from any source via Open Sound Control (OSC). It also communicates model outputs—indicating the single best current label or the posterior distribution over labels—via Open Sound Control [9]. This allows the Wekinator to be “plugged into” an existing composition or performance environment, such as Max/MSP or Processing, where its output can be visualized or used to control sound synthesis or some other process.

![Figure 1](http://code.google.com/p/wekinator/)  
**Figure 1**. The posterior distribution visualization. The position of each vertical slider indicates the classifier’s estimated posterior probability that each corresponding label is correct, given the current bow features.

2.3. Feature Extraction
The K-Bow is shipped with a software suite, K-Apps, which receives sensor values from the bow. This software provides a GUI interface for sensor calibration and debugging. K-Apps performs real-time scaling of sensor values (e.g., into the range 0–4095), and it provides infrastructure for sending sensor values to other software programs via OSC or MIDI. In our work, we used K-Apps for calibrating the bow sensors and scaling their outputs.

In order to build articulation classifiers capable of applying labels to gestures performed on the K-
Bow, it was necessary to extract feature vectors from the raw K-Bow sensor outputs. We implemented a feature extraction module that computes the following features: each K-Bow sensor’s mean, minimum, and maximum value over a sliding window; the mean, minimum, and maximum of the first- and second-order differences within the same window; and the raw sensor value sampled once per window. In total, this provided 80 features, which were sent to the Wekinator along with a metadata ID tag (not used to train the classifier).

The choice of these features is informed by prior work showing that the minimum and maximum of “velocity” and “acceleration” of sensor outputs was useful in building accurate articulation classifiers [7]. The feature extraction module runs in a GUI (Figure 2) that allows a user to select which of these available features are sent to the Wekinator and to adjust the window size and hop size. The chosen features are sent to the Wekinator via OSC. Additionally, to aid in debugging, the GUI displays the current value of each K-Bow sensor.

3. CONSTRUCTING CLASSIFIERS

3.1. The Classification Problem

We constructed a model capable of classifying seven standard bow articulations: legato (smooth and connected), marcato (onsets emphasized and slightly detached), spicato (enunciated and percussive), riccocet (a “bouncing series of 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 classifier was trained to recognize the articulations performed by a single, specific cellist who has been professionally trained (Schedel). Each training example was constructed from a single feature vector output by the feature extractor described above, which described the bow state using 80 features for a single 100ms window in time. Additionally, the input features were not segmented or filtered with attention to note onsets or bow direction changes. As a result, the classifier behavior is to assign an articulation label to the most recent 100ms of gesture, at a rate specified by the feature extractor.

3.2. The Interactive Machine Learning Process

To create training examples, the cellist specified the articulation label using the Wekinator GUI then demonstrated examples of the corresponding bow stroke. She used a foot pedal to stop and start creation of training examples. For each stroke she varied the string, bow position, bow pressure, and bow speed. After training a model from a dataset, the cellist evaluated the model by demonstrating different articulations and observing the model’s output through one or more visualizations, such as the one shown in Figure 1.

The cellist iteratively modified the articulation model by changing aspects of the data, features, or algorithm, then re-training and re-evaluating the model. She worked until she had a classifier that she liked, and overall, she created five models in succession until she was satisfied. In between subsequent re-trainings, she added new examples on five different occasions, manually edited existing examples on one occasion, deleted a subset of examples twice, deleted all examples and rebuilt the training set from scratch twice, and changed the learning algorithm, algorithm parameters, and selected features once each. The cellist was able to rapidly iterate between training, evaluating, and modifying models, in part because the model training process was relatively short (median time 4.6 s).

Iteration was crucial in enabling the cellist to fix classifier mistakes. Many of the classifier mistakes arose from the cellist having insufficiently varied dimensions of the training data that were supposed to be independent of articulation class (e.g., bow speed); in order to correct these mistakes, she added new training examples for which the dimension was more carefully varied across examples of different classes. Another type of mistake involved the classifier consistently mislabeling riccocet articulations as spicatto. Upon noticing this behavior, the cellist realized that she was actually not lifting her elbow enough on the D-string to produce a proper spicatto, so her training examples for spicatto and riccocet were too similar. After adjusting her playing technique, she was able to create clearer training data that produced a more accurate classifier.

Ultimately, the model the cellist liked best was built using a support vector machine classifier with a
polynomial kernel of degree 2, trained from all 80 available features, with a window size of 100 ms.

3.3. Creating Good Classifiers Quickly
The cellist rated her final, best classifier as having a subjective quality of “9” on a scale from “1” (very bad) to “10” (very good). Evaluating this classifier by computing 10-fold cross-validation accuracy on the training set yielded a score of 98.8%, which is comparable to or better than performance reported by prior bow gesture classification research. (Note that this score is not directly comparable, since prior work examined different numbers and types of articulation classes and performers.) Most importantly, the cellist felt that this classifier was accurate enough to be useful to her in creating an interactive composition that relied on accurate real-time bow gesture classification.

Before creating this articulation classifier, we had constructed the feature extraction module and trained classifiers for several easier bow gesture classification tasks (e.g., up bow versus down bow). The cellist had extensive experience using the K-Bow and moderate experience using the Wekinator. The process of creating the first classifier iteration, which the cellist rated “7” out of “10” in quality, took just 13.0 minutes. The entire classifier building process, from creating the first dataset to evaluating the final classifier, took 50.8 minutes.

4. DISCUSSION
The Wekinator enabled the cellist to create a working bow articulation classifier without programming or possessing machine learning expertise. The interactive machine learning process enabled her to iteratively create, evaluate, and improve the gesture model through addressing deficiencies in the training data, improving her performance technique in the creation of training examples, and experimentally altering the chosen learning algorithm and features. In the context of the larger project, the same 80 features computed by the feature extractor were also useful to the cellist in creating her own models of simpler bow gestures, such as bow direction, position, and speed. While the extent to which the gesture classifiers trained by this cellist would work accurately for other performers is unknown, we believe that other end users with similar expertise could successfully build their own gesture recognition systems using the Wekinator and these features.

In future work, we plan to create a K-Bow-specific implementation of the Wekinator that K-Bow users can download and use to build their own classifiers. This software will come packaged with the bow feature extractor and may come configured with preset algorithm and feature selection choices that we have found to be useful in classification of common bow gestures. We would also like to collect data from the bowing gestures of a variety of string players, to assess the extent to which a model trained on one performer would generalize to another performer using these sensors and features. Data from multiple performers could also be used to train a more general suite of bow gesture models that could be distributed with the software and optionally customized by end users adding their own training examples to the datasets.

5. PROPOSED DEMO
For our demo, we will invite attendees to use the K-Bow and an electric cello to train their own bow gesture classifiers. We will guide them through providing training examples of a few different gestures, training a model, evaluating its performance by observing visualizations such as Figure 1, and improving the model by changing the training data. We will also demonstrate the use of gesture classifiers trained by participants or by ourselves to drive interactive animations created by an artistic collaborator using Processing. These particle-model animations create visuals that are analogous to and contrast with the movement of the bow. For participants who are not string players, we will guide them through training gesture classifiers for nonstandard gestures, such as drawing numbers in the air using the bow. Based on our prior experience creating models for the K-Bow, we are confident that participants will be able to create working models for articulation and other gestures within a few minutes, with our guidance.

6. REFERENCES