Real-time Human Interaction with
Supervised Learning Algorithms for
Music Composition and Performance

Rebecca Anne Fiebrink

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Perry R. Cook

January 2011
Abstract

This thesis examines machine learning through the lens of human-computer interaction in order to address fundamental questions surrounding the application of machine learning to real-life problems, including: Can we make machine learning algorithms more usable? Can we better understand the real-world consequences of algorithm choices and user interface designs? How can we devise more effective human workflows for building machine learning systems, enable more successful application of algorithms by machine learning novices, and ultimately make it possible in practice to apply machine learning to new problems?

The scope of the research presented here is the application of supervised learning algorithms to contemporary computer music composition and performance. Computer music is a domain rich with computational problems requiring the modeling of complex phenomena, the construction of real-time interactive systems, and the support of human creativity. Though varied, many of these problems may be addressed using machine learning techniques, including supervised learning in particular. This work endeavors to gain a deeper knowledge of the human factors surrounding the application of supervised learning to these types of problems, to make supervised learning algorithms more usable by musicians, and to study how supervised learning can function as a creative tool.

This thesis presents a general-purpose software system for applying standard supervised learning algorithms in music and other real-time problem domains. This system, called the Wekinator, supports human interaction throughout the entire supervised learning process, including the generation of training examples and the application of trained models to real-time inputs. The Wekinator is published as a freely-available, open source software project, and several composers have already employed it in the creation of new musical instruments and compositions.
This thesis also presents work utilizing the Wekinator to study human-computer interaction with supervised learning in computer music. Research is presented which includes a participatory design process with practicing composers, pedagogical use with non-expert users in an undergraduate classroom, a study of the design of a gesture recognition system for a sensor-augmented cello bow, and case studies with three composers who have used the system in completed artistic works.

The primary contributions of this work include (1) a new software tool allowing real-time human interaction with supervised learning algorithms; (2) a better understanding of the consequences of the user interface and interaction affordances in the application of supervised learning in real-time and creative problem domains; (3) a greater insight into the roles that human-computer interaction—encompassing both human-computer control and computer-human feedback—can play in the development of supervised learning systems, (4) a clearer characterization of the human-computer interaction requirements of practicing composers and instrument designers; and (5) increased understanding of how supervised learning can function as a creativity support tool. This work both empowers musicians to create new forms of art and contributes to a broader HCI perspective on machine learning practice.
Contents

Abstract .................................................................................. iii
List of Tables ........................................................................ vii
List of Figures .......................................................................... viii

1 Bow Gesture Recognition ...................................................... 1
  1.1 Introduction .................................................................... 1
  1.2 Background and Motivation ........................................... 3
    1.2.1 Bowing Gesture Classification ................................. 3
    1.2.2 The K-Bow ............................................................ 5
    1.2.3 Motivation and Research Goals ............................... 7
  1.3 Preliminary Project ........................................................ 10
  1.4 Method ........................................................................... 12
  1.5 Observations ................................................................... 14
    1.5.1 Interactions with the Wekinator .............................. 14
    1.5.2 Bowing Gesture Models ......................................... 19
    1.5.3 Techniques and Criteria for Model Evaluation .......... 19
    1.5.4 Human Learning and Adaptation ............................ 24
    1.5.5 Cellist’s Final Evaluation of the Wekinator and Models 27
  1.6 Discussion ....................................................................... 28
    1.6.1 Subjective Ratings and Cross-Validation ................ 28
    1.6.2 Efficacy of Interactive Supervised Learning with the Wekinator 30
List of Tables

1.1 K-Bow Sensor Summary ............................................. 8
1.2 Bow Gesture Classification Tasks .............................. 13
1.3 Bow Gesture Classification Model Performance .......... 21
List of Figures

1.1 K-Bow Sensors ........................................... 7
1.2 K-Bow Emitter ........................................... 7
1.3 K-Bow Feature Extraction Application ..................... 12
1.4 User Actions, Session A .................................. 16
1.5 User Actions, Session B .................................. 17
1.6 Summary of User Actions ................................ 18
1.7 Training Set Sizes and Times ............................... 20
1.8 Posterior Visualization Tool ............................... 24
1.9 Models’ Cross-Validation and Subjective Rating .......... 29
Chapter 1

Bow Gesture Recognition

1.1 Introduction

In this chapter, we discuss work undertaken with a professional cellist/composer to build a gesture recognition system for a commercially-produced, sensor-equipped cello bow. This bow, called the “K-Bow,” contains embedded sensors for measuring the position and motion of the bow in real-time (McMillen 2008). One goal of this work was to build a gesture classification system for standard cello bowing gestures, such as bow direction (“up-bow” or “down-bow”) and articulation (e.g., “legato,” “marcato,” and “spiccato”), for use in composition and live performance. A second goal of this work was to investigate how interactive machine learning can be useful for building gesture classifiers for discriminating among a pre-defined set of musical gestures. We sought to discover the subjective criteria employed by the cellist to evaluate trained models, how interactive model evaluation and training data editing enabled her to improve the models, and the ways that she was educated and influenced by her interactions with the software.

In this work, we have employed a user-centered design approach to the collaborative creation of the classifier suite and supplemental software infrastructure, in
conjunction with an observation-analysis approach to discovering how the cellist employed the Wekinator to evaluate models, create training data, and refine her strategies for effective model-building. We begin this chapter with an overview of related work on bowing gesture analysis, a description of the K-Bow, and a more thorough discussion of our research motivation and goals. We describe our research method in detail and present the findings of our collaboration with and observation of the cellist/composer. Specifically, we present our observations regarding how interactions with the Wekinator were used in model-building, the quality of the models produced, the cellist’s techniques and criteria for model evaluation, the ways in which the Wekinator drove the cellist’s own learning and adaptation, and the cellist’s overall evaluation of the Wekinator and models. We then briefly discuss the findings of this work with regard to the subjective and objective evaluation of supervised learning systems, the efficacy of applying interactive supervised learning with the Wekinator to bow gesture classification, and proposed future improvements to the Wekinator software. We further discuss the implications of this work regarding interactive machine learning applications, interfaces, algorithms, and evaluation methods in Chapter ??.

In the work presented in this chapter, we found that the cellist was able to successfully employ the Wekinator to create bow gesture models that were of a sufficiently high quality to use in performance. She employed an interactive, iterative approach to building most of these models, in which she alternated between evaluating models and taking action to improve them by changing the training set, feature selection, or learning algorithm. We found that the cellist employed a variety of criteria for assessing the quality of a trained model, beyond just an assessment of its correctness; furthermore, the cellist’s subjective rating of a model’s quality did not always positively correlate with its cross-validation accuracy. These findings raise questions regarding the extent to which cross-validation and other accuracy measures are useful evaluation metrics in interactive contexts. We also found that interacting with
the Wekinator led the cellist to become a more effective user of interactive machine learning, enabled her to redefine her goals for the models over time to account for what the models were able to learn well, and led her to gain a new perspective on her own bowing technique.

1.2 Background and Motivation

1.2.1 Bowing Gesture Classification

The problem of identifying and analyzing standard violin bow strokes by applying supervised learning to analyze bow sensor data has been studied previously (Peiper et al. 2003; Rasamimanana et al. 2005; Young 2008). This prior work focuses specifically on classifying different bowing articulation techniques, where articulation is defined as 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…” (Slonimsky 1998, 16). 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 Flesch (2000).

Peiper et al. (2003) built a bow position sensor system using a pair of electromagnetic field sensors, then studied the application of decision tree classifiers to discriminate among subsets of standard bow strokes (martelé, détaché, spiccato, legato, and staccato). They designed the feature extraction system so that each feature vector characterized the bow position, speed, and acceleration for a complete bowing action in a single direction (i.e., an up-bow or a down-bow). The dataset for this work was collected by recording sensor outputs during a human demonstration of different bow strokes, then manually annotating the dataset with the proper bow labels; the design of the dataset (e.g., the proportions of each class, and the string or dynamic level
used to demonstrate each stroke) is not otherwise specified in the published work. The decision trees obtained accuracy in the range of 71% (discriminating among all five bowing classes) to 100% (discriminating between only détaché and martelé) on the data.

Rasamimanana et al. (2005) more thoroughly investigated the effects of features, performer, dynamic level, and tempo on the classification of martelé, détaché, and spiccato strokes. For the measurements, they attached position and acceleration sensors to a standard violin bow. The data used for experimentation was collected from two violinists who were directed to play each articulation while varying the violin string, dynamic level, and tempo in a prescribed manner. It was observed that the maximum and minimum acceleration and velocity in the direction of the bow movement were the features most predictive of the articulation style. In applying a k-nearest neighbor algorithm to the three-stroke classification problem on the data collected from the two performers, Rasamimanana et al. achieved per-class classification accuracy in the range of 85.8% to 96.7% on a held-out test set.

Most recently, Young (2008) applied a k-nearest neighbor algorithm to discriminating among six articulations: accented détaché, détaché lancé, louré, martelé, staccato, and spiccato. The bow used in this work was a standard violin bow specially outfitted with sensors for measuring downward and lateral force, three axes of acceleration, and angular velocity around these three axes. Data was collected from eight violinists playing an musical excerpt demonstrating each articulation on every string, and playing at a single dynamic level and a fixed tempo set by a metronome. In this work, each data instance used for training and classification was comprised of features extracted over a set of sequential, tempo-controlled bow strokes of the same type; that is, the goal of the classification was not to provide a note-level or instantaneous classification of articulation. Applying a standard dimensionality reduction technique to the
dataset followed by a k-nearest neighbor classifier resulted in per-class classification accuracies of 91.7% to 97.9%, computed by three-fold cross-validation.

The dataset creation and evaluation methodologies employed in this prior work, especially in the work by Rasamimanana et al. and Young, have treated the problem of bow gesture classification as a conventional machine learning problem. The datasets were carefully designed, with certain bowing characteristics (such as tempo) intentionally held fixed in order to create a simpler problem than that of classifying realistic performance-time gestures, while other bowing characteristics (such as which of the four instrument strings were played) were intentionally varied in order to create classifiers that were robust to these variations. Having created a dataset that was thus representative of the researchers’ chosen scope of the problem, the goal of algorithm and feature selection was to model the dataset as faithfully as possible, and to measure success using test set or cross-validation accuracy. Therefore, the evaluations performed in this prior work are not directly informative of how accurately the classifiers would classify bow gestures used in an actual performance, nor how useful they would be for a composer desiring to incorporate them into a composition. However, these results do indicate that supervised learning is a promising technique for classifying articulatory bow gestures from these types of sensors and features.

1.2.2 The K-Bow

The K-Bow is the first commercially-developed, mass-produced sensor bow for string players (McMillen 2008). It contains several sensors, shown in Figure 1.1, for measuring the position and motion of the bow in real-time. A three-axis accelerometer located inside the frog (i.e., the large, rectangular assembly at the bottom left of Figure 1.1 located at the end of the bow held by the player) 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 at
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, shown in Figure 1.2 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 1.1.

The K-Bow is manufactured in versions for violin, viola, cello, and bass. Each version of the bow is designed to allow the string player to play using standard technique without encumbrance by the sensors, and this is accomplished through a wireless setup and through the bow’s physical construction, which is designed so that the size and distribution of weight throughout the bow match a standard instrument bow. The power source and circuitry for the on-bow sensors are located inside the frog, and the sensor values are wirelessly communicated to a computer up to 10m away via Bluetooth. The published data rate for the Bluetooth transmission or sensor values is up to 625Hz, and we observed similar data rates in our work.

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, for example to allow the musician to check that the Bluetooth connection is still alive and all bow sensors are working properly. 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, as well as mapping sensor values directly to controlling K-Apps’ built-in modules for audio spatialization, sample looping, and other musical processes. In our work, we used K-Apps only for sensor calibration and scaling, debugging, and sending sensor values to our own software via OSC.
1.2.3 Motivation and Research Goals

One goal of this work was to use the Wekinator to construct a set of robust classifiers for standard cello bowing gestures that the cellist/composer could use in composition and performance. As a professional computer music composer, she often creates interactive computer music compositions in which human actions trigger or dynamically...
Table 1.1: A summary of K-Bow sensors.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Bow acceleration and tilt, measured by internal 3-axis accelerometer</td>
</tr>
<tr>
<td>y</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td></td>
</tr>
<tr>
<td>hair (h)</td>
<td>Bow hair tension</td>
</tr>
<tr>
<td>grip (g)</td>
<td>Grip pressure and surface area</td>
</tr>
<tr>
<td>length (l)</td>
<td>Horizontal distance between frog and fingerboard</td>
</tr>
<tr>
<td>bridge (b)</td>
<td>Vertical distance between bow and bridge</td>
</tr>
<tr>
<td>tilt (t)</td>
<td>Tilt of bow relative to instrument</td>
</tr>
</tbody>
</table>

influence sound and visuals produced by the computer. Additionally, she often participates in the performance of her own compositions, playing the cello with the K-Bow as well as as manipulating the computer directly through a GUI. Having access to a set of real-time bow gesture classifiers would enable more natural performance-time interactions with the computer; in the words of the cellist, “It allows me to augment the bowing skills I spent years working on.” These classifiers could enable her to trigger and control computer processes using her existing repertoire of bowing techniques, and potentially enable the computer to track the position of the cellist as she plays through a notated score. Additionally, bow gesture classifiers can reshape the process of composition, allowing her to focus on how the musical meaning of bowing gestures—rather than the sensor values themselves—might affect the evolution of a piece. This presents clear practical benefits: “The Wekinator makes the compositional/programming process faster—its something I’ve wanted to do for a long time, but the amount of data was daunting.” Finally, incorporating computer understanding of bowing gestures also has aesthetic appeal to her as a composer, as “the bow is really where the expression in a string instrument lies.”

A second goal of this work was to investigate human-computer interaction with supervised learning algorithms in a gesture classification task that was more constrained than those explored in the previous two chapters. While the composers in Chapter ?? and the students in Chapter ?? had some leeway in choosing and changing the
gestural vocabulary and the gesture-sound relationships they wanted the Wekinator models to learn, the learning problems addressed in our work with the K-Bow were constrained to modeling a gestural vocabulary that was fixed according to musical conventions. Furthermore, while the users studied in the previous two chapters were able to adapt their own performance-time gestures in order to elicit certain model behaviors, the goal in this work was to create classifiers that worked with the cellist’s existing bowing technique, without any adaptation on her part. The aspects of human interaction that we were most interested in were the evaluation criteria and techniques used by the cellist to evaluate bowing gesture models, the ways that interaction with the Wekinator enabled her to improve the models, and the ways that the cellist was educated and influenced by the interactive machine learning process.

This work complements previous research on bowing gesture classification in that it is focused on the process and techniques through which people—perhaps the performers themselves—can build gesture classifiers that are most accurate and useful. Prior work has focused on designing and evaluating supervised learning systems that model a carefully pre-defined dataset as accurately as possible, without attention to how the models will be used in practice, and without engaging musicians’ interaction or expertise beyond the initial creation of the fixed training set. In contrast, the goal here is to enable the musician who will be performing with the models to apply her musical expertise to making them as useful as possible. As we will demonstrate, this entails both enabling her to evaluate the models based on her own criteria for usefulness (which, as it turns out, includes more than just classification accuracy), and allowing her to take action to improve the models against these criteria.
1.3 Preliminary Project

We conducted a preliminary project with the cellist/composer and with the software engineer of K-Apps to develop and refine infrastructure for computing features from the raw bow sensor outputs and communicating these features to the Wekinator. In parallel, we collaborated with the cellist to teach her how to use the Wekinator, solicit her feedback on its user interface and design, and experimentally apply it to classifying several standard bowing gestures. These classification problems, which are further explained in Table 1.2, included bow direction, position on or off the strings, speed, vertical position, horizontal position, roll, and articulation. We aimed to demonstrate that the standard classification algorithms and interactive supervised learning process supported by the Wekinator—in particular, the use of training sets created through a few minutes of unstructured, interactive demonstration—were sufficient to allow a K-Bow user to construct her own working classifiers. We also wanted to discover which methods for segmenting and extracting features from the sensor outputs were viable, given that previous work on bow gesture classification had employed different sensors and a variety of segmentation methods. Additionally, we sought to discover which improvements to the Wekinator software were necessary to better support this type of classification task, and to implement them. This work was presented at the Third International Conference on Music and Gesture (Fiebrink et al. 2010).

Our working process in this preliminary project was unstructured and exploratory. We met with the cellist in person on four occasions, for several hours each time, over the course of six months. During each meeting, we showed the composer how to use the most current version of the Wekinator, tested our current feature extraction and communication software, experimented with building gesture classifiers using different algorithms and features, and discussed how the Wekinator might be improved to make the classifier-building process easier. In between meetings, we further developed the Wekinator and feature extraction software based on our experiences and discussion.
We were successful in building proof-of-concept classifiers for each of the seven bow gesture classes listed above. The cellist did not rigorously evaluate the classifiers, but in each of the seven problems, she assessed that the classifier’s accuracy was adequate and that she was confident she could address remaining significant errors in its performance through further refinement of the training data or algorithms within the Wekinator. Computing cross-validation accuracy of these classifiers yielded scores in the range of 80% to 100%.

This project led to several features being added to the Wekinator software. Most notably, the graphical interface discussed in Section ?? was created for visualizing and editing the training data. By using this interface to manually add class labels, the cellist could use more natural performative gestures to create the training data, switching fluidly between gesture classes (e.g., up-bows and down-bows) without pausing to interact with the GUI. This interface was also useful in cleaning up the training data, for example removing ambiguous instances recorded while the cellist was switching from an up-bow to a down-bow.

This project also led to the development of a K-Bow feature extraction application, external to the Wekinator, which computed features found to be useful for bow gesture classification. This application’s GUI is shown in Figure 1.3. The values of the eight bow sensors are sent to this application from K-Apps, and it computes the feature vector and sends it to the Wekinator using OSC. For each of the bow sensors, this application computes the average, minimum, and maximum of the sensor value, and of the first- and second-order differences of the sensor value, calculated over a sliding analysis window whose size is specified by the user. The software also samples the raw sensor values once per window. The GUI allows a user to select which of these available features are sent to the Wekinator, adjust the window size, and adjust the rate at which features are computed and sent (i.e., the “hop size”). Furthermore, to aid in debugging, the GUI displays the current value of each K-Bow sensor output.
1.4 Method

Having established the Wekinator as a suitable tool for building bow gesture classifiers, we conducted a more formal study of the application of interactive supervised learning to the construction of performance-ready classifiers. In this study, conducted five months after the preliminary project, we sought to address our research questions through observing and analyzing the cellist’s actions and comments as she worked to construct eight usable classifiers for the gestures in Table 1.2.
Table 1.2: Bow gesture classification tasks.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction</td>
<td>The direction the player is bowing, e.g., up-bow, down-bow, neither</td>
</tr>
<tr>
<td>On/Off String</td>
<td>Whether or not the bow is in contact with one or more strings of the instrument</td>
</tr>
<tr>
<td>Grip</td>
<td>Whether or not the cellist was squeezing the grip sensor</td>
</tr>
<tr>
<td>Roll</td>
<td>Whether or not the bow was rolled with the edge of the hair against the strings (to play more quietly) or positioned normally, with the hair more flat against the strings</td>
</tr>
<tr>
<td>Horizontal Position</td>
<td>The horizontal position of the bow relative to the instrument, i.e., whether the frog, middle, or tip of the bow is in contact with the strings</td>
</tr>
<tr>
<td>Vertical Position</td>
<td>The vertical position of the bow relative to the instrument, i.e., sul tasto (bow over the fingerboard), sul ponticello (bow near the bridge), or neither (bow in the middle)</td>
</tr>
<tr>
<td>Speed</td>
<td>The speed of the bow against the strings, e.g., “Very slow” to “Very fast,” according to the cellist’s own definitions of these terms</td>
</tr>
<tr>
<td>Articulation</td>
<td>The bowing technique employed to affect notes’ onsets, releases, and transitions, including: legato (smooth and connected), marcato (onsets emphasized and slightly detached), spiccato (“enunciated” and percussive), ricocet (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) (see Flesch (2000) for further discussion)</td>
</tr>
</tbody>
</table>

The process of applying the Wekinator to creating these classifiers was collaborative. We assisted the cellist with the machine learning component of the work, discussing the algorithms, parameters, and features with her. She was responsible for assessing the quality of the trained classifiers and creating training data, and we did not assist in these tasks. We recorded the cellist’s actions and comments throughout this process using a combination of written notes, video, and automated logging. The Wekinator logged all interactions with the software, and it saved all models, all training data, and all data generated as the cellist ran trained models to classify new bow gesture inputs. Additionally, we asked the cellist to evaluate and verbally rate the
quality of each trained model on a 10-point scale. We recorded these ratings along with her commentary explaining the scores, if any.

For each of the eight gesture classification tasks, the cellist started from an empty training set and built a classifier from scratch (i.e., the training data and models from the preliminary study were not used). She worked on improving the classifier for each task until she was satisfied with its performance or felt she could not improve it further. The cellist freely chose the order in which the classification tasks were addressed and the amount of time to allocate to each one. A secondary set of classifiers were built in a separate, later session of model-building for five of the more difficult tasks, in order to allow the cellist to use her knowledge gained from building the first set of models. For clarity in the following discussion, we will refer to the first session of model-building, in which models were built for all eight classification problems, as “A,” and to the second session, in which models were re-built from scratch for five classification problems, as “B.”

1.5 Observations

1.5.1 Interactions with the Wekinator

In building the Vertical Position, Grip, and Roll classifiers in Session A, the cellist did not employ an iterative approach to model building. After creating each classifier and directly evaluating it, she rated their subjective performance as “10” and judged that they did not require further improvement. These three classification problems were among the most straightforward of the tasks: each had only two or three classes, which were easy for the cellist to demonstrate unambiguously and which depended on quite simple properties of the K-Bow sensor features. Iteration was not needed because the initial set of training examples clearly represented the problem, and the
initially-chosen classifier algorithm was able to create an accurate model from the data.

In building the other five classifiers in Session A, the cellist employed an iterative approach to model building. She identified problems with a model through evaluating it (typically by running it on new gestures in real-time, as discussed in Section 1.5.3), then addressed these problems through modifications of the training data, algorithm, and/or features. Figure 1.4 illustrates the actions taken to build the models in the five classification tasks in A for which the model was trained more than once. These actions are summarized in Figure 1.6a, which displays the total and per-task average of the number of times each system interaction occurred.

In building each of the eight classifiers in Session A, the learning algorithm was retrained an average of 3.7 times ($\sigma = 6.8$, median = 1.0) (not counting the first training). Each re-training followed a modification of the training set, learning algorithm, and/or selected features, and each of these actions was important in at least one task. Among the five tasks for which the classifier was retrained at least once, the training data was modified somehow after its initial creation an average of 2.5 times (median = 1.5, $\sigma = 3.1$), the learning algorithm and/or its parameters were changed an average of 2.8 times (median = 1.5, $\sigma = 4.1$), and the selected features were changed an average of 2.3 times (median = 1.0, $\sigma = 3.8$).

In the second session of model building, B, the learning algorithm was retrained an average of 1.0 times (median = 1.0, $\sigma = 1.0$) during each of the five tasks, after the initial training. Again, editing the training data, algorithm, and features each played a part in the process of improving the models. Figure 1.5 illustrates the actions taken to build the models in the three classification tasks in B for which the model was trained more than once. Figure 1.6b summarizes the total and per-task average number of occurrences of each system interaction in B.
Figure 1.4: Actions taken for each task in A for which the model was re-trained at least once. Each action appears above the training iteration that it precedes; e.g., between the first and second trainings of the Horizontal Position model, the model was directly evaluated and the algorithm parameters were changed. Actions above the red “Stop” × indicate actions taken after the last model training, before the cellist decided to stop building the model.
Figure 1.5: Actions taken for each task in B for which the model was re-trained at least once. See Figure 1.4 caption for more information.
Figure 1.6: Summary of the number of times each action was performed in between re-trainings of the model. Here, if an action was performed more than once between subsequent re-trainings, it is only counted once.

The cellist iteratively modified the training set, algorithm, and features in predictable ways as she worked to improve the models. To correct misclassifications occurring for a particular type of gesture, she would often add more examples of that gesture to the training set, with the proper labels. When she felt her training set was well-representative of the problem, but classification accuracy was still poor, the algorithm, its parameters, or its features were changed in an attempt to create a better model of the current data. Occasionally the algorithm parameters were changed to
address very specific goals, such as increasing the value of $k$ for k-nearest neighbor to make the model’s labels more consistent through small changes in gesture.

In general, the attempted improvements to the model were effective, in that they resulted in a subjectively improved model rating following retraining. The average increase in rating from one iteration to the next was 0.48 (median= 0.5, $\sigma = 2.1$), and the average increase in model rating from the first iteration of the problem to the final iteration was 2.7 (median= 2.0, $\sigma = 2.0$).

The training datasets were small enough that the training process did not interrupt interaction with the system. Figure 1.7 shows the size of the average and largest training dataset for each classification task, and the average and maximum training time per task, for all trainings in A and B. Over all, the average model training took 4.4 seconds (median= 0.2, $\sigma = 17.7$). In total, 204.9 minutes were spent interacting with the Wekinator to build the eight classifiers in A, and 44.4 minutes were spent to build the five classifiers in B.

### 1.5.2 Bowing Gesture Models

The best-rated models developed for each classification problem are described in Table 1.3. As the table shows, it was possible to construct models rated highly by the cellist (as “9” or “10”) for all classification problems. The cross-validation accuracy computed on the training set of each of these models is also moderately high, with a minimum of 83.5% and an average of 91.8%.

### 1.5.3 Techniques and Criteria for Model Evaluation

Many of our recorded observations and logging concerned the ways in which the cellist evaluated trained models, the criteria she used to assess a model’s quality, and the ways her evaluations informed her interactions with the system. In this section, we discuss our observations with regard to the two evaluation techniques employed:
cross-validation, and running the trained model on new gestural inputs in real time, which we will refer to as direct evaluation.

**Cross-Validation**

Cross-validation was used only in Session A, and it was computed 14 times in total: once for On/Off, once for Direction, once for Articulation, and 11 times for Speed. Cross-validation was used for the Speed, Articulation, and Direction tasks after direct evaluation had revealed the learning problem to be particularly stubborn, and several different algorithms and feature selections were tried in succession to see if any one
Table 1.3: The models rated by the cellist as the best for each classification task, built in the first model-building session, “A.” The columns, in order, include: the task, the cellist’s rating (on a 1–10 scale), the number of training iterations performed before producing this model, its 10-fold cross-validation accuracy, the number of discrete classes, the classification algorithm, and the features used by the classifier. The features are computed by the feature extraction application described in Section 1.3, computed from the base features listed in Table 1.1.

<table>
<thead>
<tr>
<th>Task</th>
<th>Rating</th>
<th>Iter.</th>
<th>CV</th>
<th>Classes</th>
<th>Classifier</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction</td>
<td>10</td>
<td>5</td>
<td>87.3</td>
<td>4</td>
<td>kNN</td>
<td>$l_{vmin}$, $l_{vmax}$, $l_{vmean}$, $l_{amin}$, $l_{amax}$, $l_{amean}$;</td>
</tr>
<tr>
<td>On/Off String</td>
<td>10</td>
<td>2</td>
<td>83.5</td>
<td>2</td>
<td>AdaBoost on J48</td>
<td>$h_{mean}$</td>
</tr>
<tr>
<td>Grip</td>
<td>10</td>
<td>1</td>
<td>100</td>
<td>2</td>
<td>kNN</td>
<td>$g_{mean}$</td>
</tr>
<tr>
<td>Roll</td>
<td>10</td>
<td>1</td>
<td>98.2</td>
<td>2</td>
<td>AdaBoost on J48</td>
<td>$min$, $max$, and mean of $x$, $y$, and $z$; $b_{min}$, $b_{max}$, $b_{mean}$;</td>
</tr>
<tr>
<td>Horizontal Position</td>
<td>10</td>
<td>2</td>
<td>89.3</td>
<td>3</td>
<td>kNN</td>
<td>$l_{raw}$, $l_{mean}$</td>
</tr>
<tr>
<td>Vertical Position</td>
<td>10</td>
<td>1</td>
<td>90.0</td>
<td>3</td>
<td>J48</td>
<td>$b_{raw}$, $b_{mean}$</td>
</tr>
<tr>
<td>Speed</td>
<td>9</td>
<td>21</td>
<td>87.5</td>
<td>5</td>
<td>AdaBoost on J48</td>
<td>$l_{vmin}$, $l_{vmax}$, $l_{vmean}$, $l_{amin}$, $l_{amax}$, $l_{amean}$;</td>
</tr>
<tr>
<td>Articulation</td>
<td>9</td>
<td>5</td>
<td>98.8</td>
<td>7</td>
<td>SVM</td>
<td>all</td>
</tr>
</tbody>
</table>

might result in a usable model. Cross-validation was convenient for this purpose because it provided a faster and more consistent way of comparing models than direct evaluation: each round of cross-validation took, on average, 1.1 seconds ($\sigma = 1.5$).

**Direct Evaluation**

When the cellist wanted to assess the performance of a model, to decide whether the model performed satisfactorily and she could start a new classification task, or to assess how the model might be improved, she ran the trained model as she demonstrated bowing gestures in real-time. During this direct evaluation of a model, the
The cellist had the option of displaying its outputs textually in the Wekinator GUI or using them to control a live visualization, as discussed below.

Because the cellist was asked to assign the current model a subjective rating from 1 to 10 based on her direct evaluation, this entailed a minimum of one direct evaluation per classification task. It took her an average of 44.1 seconds of hands-on evaluation to make a rating judgement (median = 35.5, $\sigma = 31.3$). On average, she spent 52.8 seconds (median = 36.9, $\sigma = 52.2$) in each round of direct evaluation, and in total, 38.3 of 249.2 minutes with the system were spent directly evaluating the models. The cellist directly evaluated models an average of 5.4 times per task in A (median = 1.5, $\sigma = 7.6$) and an average of 2.6 times (median= 3, $\sigma = 1.7$) per task in B.

The cellist’s verbal comments during direct evaluation and her choices to act to correct certain aspects of model behavior indicated that her criteria for assessing a model’s quality included its correctness, subjective cost or severity of errors, decision boundary shape, and posterior probability distribution shape. We elaborate on these criteria in the next sections.

**Correctness and Cost**

In all bowing classification tasks except for Speed, the categories being learned by the model were defined by musical convention. (In the Speed task, the cellist herself designated certain ranges of speed as “fast,” “very fast,” etc.) When a model’s classifications output during direct evaluation deviated from what was musically correct, this was predictably judged by the cellist to be an incorrect model behavior. This behavior was most often addressed by adding additional training examples to the training dataset, which were similar to those that caused the misclassification, but with the correct labels. In extreme cases of failure, the cellist recreated the training set from scratch. The cellist did indicate that different misclassifications had different degrees of severity, based on the musical appropriateness of the classifier’s
label. For example, classification mistakes a human cellist might easily make were less problematic.

**Decision Boundary Shape**

The cellist occasionally complained that, as she gradually changed from one bow gesture to another, a classifier’s output might jump around unpredictably before stabilizing. When classifying Horizontal Position, for example, it was very important to her that the classifier cleanly switch from a label of “frog” to a label of “middle” at some point during a down-bow stroke, rather than jump between the two; it was less important that this label switching happen at a precise horizontal position. In other words, the shape and smoothness of the classifier decision boundaries in the gesture space were more important than their exact locations. Actions taken to smooth jagged decision boundaries included changing algorithm parameters (e.g., increasing $k$ in k-nearest neighbor) and adding smoothly-labeled training data along the boundary area.

**Confidence and Posterior Distribution Shape**

When evaluating a model’s quality, the cellist also took into account the shape of the model’s estimated posterior probability distribution over the class label set. The cellist was a proficient programmer, and she worked with us during the study to design several simple visualization applications to help her understand more about the model during direct evaluation. One frequently-used visualization, shown in Figure 1.8, displayed the estimated posterior distribution as it changed in real-time, using a set of sliders. When the model classified her current bow gesture correctly but also assigned relatively high posterior probabilities to several incorrect labels, the cellist expressed dissatisfaction (at one point exclaiming, “Come on, be more sure than that!”) and attempted to improve the model’s confidence. In the Horizontal Position
task, she actually changed her model rating from “10” to “9” when observing that the distribution appeared to gradually change between the “middle” and “tip” classes but not between the “frog” and “middle” classes while she bowed.

The cellist also considered the information gained from the posterior distribution as potentially helpful for improving the practical usefulness of a poorly-performing classifier. For example, when evaluating an Articulation classifier using the posterior visualization, she noticed that although the model often output the wrong label for three of the articulations, the posterior distribution for these articulations had a predictable “signature” shape that could be post-processed by some simple code to produce the correct label.

1.5.4 Human Learning and Adaptation

The cellist modified her own goals and behaviors throughout the interactive machine learning process. She remarked that her strategy for providing training data “definitely evolved over the training sessions.” By the end of the study, her strategy for classification problems she knew from experience were easier to model was to provide as varied a training dataset as possible, varying “which string, bow position (frog to tip and fingerboard to bridge), [and] speed and preparation (i.e., how high off the
string I would start)...” to make the trained model maximally robust to these effects. On the other hand, for problems that she discovered were more difficult to model, she started by simplifying the problem represented in the training dataset, keeping variables such as speed and choice of string constant across all training examples in order to build a model more likely to discriminate between classes based on only truly relevant criteria. In this, the cellist intentionally used the training set to represent the scope of the classification problem, just as was done in the prior bow classification research discussed in Section 1.2.1. However, here, this scoping was fluid, changing from problem to problem and over the course of a task, in response to both how the cellist anticipated using the model in performance as well as how well the model was performing.

There were no significant differences between model ratings in A compared to those in B; final model ratings for the five classification tasks completed in both A and B were identical, and a paired t-test comparing initial model ratings for each task yields \( p = 0.88 \). Nor was there a statistically significant difference in the number of iterations performed for each task (paired t-test yields \( p = 0.55 \)), though the average number of iterations for these tasks was 5.4 in A and 2.6 in B, due in large part to the fact that it took 21 iterations to create a satisfactory Speed classifier in A, but only 1 iteration in B. Also, the total time spent on these tasks in B was markedly less than in A (44.6 minutes in B versus 118.5 minutes in A), but this is due primarily to the reduced time spent on the Speed task, and the paired t \( p \)-value is not significant \( (p = 0.32) \). One interpretation of these results is that, while the knowledge gained in the thorough and long investigation of training data creation strategies, learning algorithms, and features undergone in building the Speed classifier in A was clearly beneficial in building the Speed classifier in B, the one or two training rounds devoted in A to each of the other four tasks (On/Off, Horizontal Position, Vertical Position, and Roll) did not provide noticeable benefits to attempting these tasks a second time.
The cellist’s goals for the models, though constrained by the need to apply musically appropriate labels to natural performer gestures, were still sometimes adjusted to reflect what a model was able to learn. For example, in the Speed task in session A, after building a Speed classifier that worked well for three classes of speeds, she decided to try building a finer-grained Speed classifier for five classes. As another example, the cellist started the Bow Direction classification task with only three classes, “up-bow,” “down-bow,” and “not moving.” However, when the initial trained models did not perform as well as expected for this very simple classification task, a fourth class was added. This class, “none of the above,” was used to represent the state of the bow when it was changing direction.

Interaction with the Wekinator also led the cellist to gain a new perspective on her own bowing technique, when occasionally she discovered through consistently poor model performance that her training data was not as clear as she thought it had been. For example, noticing that the bowing articulation model was not discriminating well between ricocet and spiccato strokes, she reexamined her own technique for those strokes and discovered that her spiccato technique actually needed to be improved in order to be less like ricocet. After adjusting her technique, she was able to both train a model that performed better and produce a better cello sound. As a second example, the cellist learned through many subsequent failures to produce a working Speed classifier in Session A that her bow speed was often inconsistent and did not match her perception of how fast she was bowing. In reality, her perception of bow speed had more to do with the speed at which she was playing notes (i.e., by moving her non-bow-hand fingers on the strings) and the speed at which she was changing bow direction (which was increased by shortening the length of the bow used in each up-bow and down-bow, not increasing the horizontal bow speed). Neither of these components of bowing speed were well-captured by the bow length sensor features being used to classify speed, so the model performed poorly. The cellist decided that
her goal for the Speed classifier was nevertheless to classify the horizontal speed of
the bow, not her perception of speed, so she trained herself to consistently move the
bow at set speeds by attaching colored markers to her bow and watching them as she
played. Through this technique, she was able to create a cleaner training set and a
model that she liked.

1.5.5 Cellist’s Final Evaluation of the Wekinator and Models

The cellist moderately to very highly agreed with each of three statements regarding
the usefulness of the Wekinator. On a 5-point Likert scale, she rated “The Wekinator
was able to create accurate bow stroke classifiers in our work so far” as a 4. She had
two responses for the rating of “The Wekinator was able to create bowing classifiers
that would be useful in performance”: indicating that “they could be used in perform-
ance, but would have to be combined with other factors in order to make [them]
truly musically relevant,” she rated her agreement as 3.5, but said that her rating
increased to 4.5 “if you simply want a bow stroke to trigger... a change.” That is, as a
composer, she considered the job of making models musically useful to encompass far
more than creating models that performed accurately; it also encompassed composing
a musically appropriate context in which the models could be used most effectively.
Finally, asked to rate her agreement with the statement “The Wekinator was able to
create bow stroke classifiers more easily than other approaches” on the 5-point scale,
she responded “10 (so 5),” indicating that the ease of creating classifiers with the
Wekinator was a key advantage of the software.
1.6 Discussion

1.6.1 Subjective Ratings and Cross-Validation

We compared the cellist’s subjective model ratings against 10-fold cross-validation accuracy computed on the same models, to assess the relationship between subjective model quality and the estimated generalization accuracy computed by cross-validation. Figure 1.9 plots rating against cross-validation accuracy (computed after the study’s completion) for all models created and rated in both A and B.

The cellist’s rating of a model was sometimes—and sometimes strongly—correlated with the estimated generalization accuracy, but this was not the case for all tasks. For each of the six classification tasks where three or more training iterations were performed, we computed the Pearson’s correlation between the cellist’s rating and the cross-validation accuracy. The Horizontal Position, Vertical Position, Bow Direction, and On/Off String classification tasks had negative correlation coefficients (−0.59, −0.44, −0.74, and −0.50, respectively), while speed and articulation classification tasks (for which there were considerably more trainings performed) had positive coefficients (0.65 and 0.93). Computing the Pearson’s correlation between cellist rating and cross-validation accuracy over all models and all tasks yields a coefficient of $r = 0.69$. Computing the Spearman’s rho rank correlation—a measure of correlation without an assumption of a linear relationship (Wilcox 2010, 178)—yields a coefficient of $\rho = 0.68$.

The finding of a negative correlation between subjective rating and cross-validation score for some tasks seems unexpected in the context of prior work in building music classification systems. For example, in all work discussed in Section 1.2.1 above, cross-validation and test accuracy are the only evaluation methods used to assess the trained models’ performance. The underlying assumption is therefore that cross-validation and test accuracy are good indicators of how well a model will
Figure 1.9: The cellist’s subjective rating and the 10-fold cross-validation accuracy score for all models created in A and B, grouped by classification task.

perform in practice. In the four tasks here with negative correlation coefficients, models with high cross-validation accuracy and a low user rating can be explained as the result of the training set providing a representation of the learning problem that was both inaccurate and too simple. For example, an initial version of the horizontal position training set contained mislabeled examples for all instances of one class. The training dataset was easy to classify correctly but the resulting model was useless. In the other tasks, this negative correlation likely resulted from the cellist unknowingly co-varying the bowing class of interest with more easily distinguishable aspects of the gesture (such as the string being played), effectively leading the model to learn the wrong concept. In all four cases, problems with the training set were undetectable using cross-validation, whereas direct evaluation allowed the cellist to discover the problems, fix them, and ultimately create models rated “10” for each task.
The sometimes negative correlation between cross-validation accuracy and subjective rating suggests that the training set may be a poor resource for estimating generalization performance during certain stages of the interactive model creation process, especially when the user has not yet discovered problems with the training data. We further discuss implications of this finding in Section ??.

1.6.2 Efficacy of Interactive Supervised Learning with the Wekinator

The high quality ratings assigned to the final models in Table 1.2 and the cellist’s overall assessment of model accuracy and usefulness presented in Section 1.5.5 indicate that these models are of a quality high enough to be used in performance. (The cellist/composer is currently working to integrate them into new compositions.) Because our work only concerned a single cellist, we do not claim that our classifier suite will generalize well to the bowing of other performers. Through future work with a larger pool of string players, we could apply this process to creating a more robust set of classifiers for wider distribution. However, a more immediate future application of this work is to produce a K-Bow-specific version of the Wekinator, with algorithms and features pre-selected for standard bowing gestures, and with an interface for K-Bow users to interactively provide and refine their own training examples.

The final Articulation model’s cross-validation score of 98.8% is better than or comparable to previously reported results for bow classification by Rasamimanana et al. (2005), Peiper et al. (2003), and Young (2008), even while it is capable of discriminating among more articulation classes than those studies. While there is still room to improve the performance of the Articulation model, and while cross-validation is a problematic metric for reasons discussed above, this and other study outcomes indicate that an interactive supervised learning approach can build models that are at least as accurate as the non-interactive, though methodical, approaches
to training data creation and model design described in the literature. In contrast to a conventional machine learning approach, the interactive process enabled the user to effectively improve the models’ performance on each task through a variety of strategies, including modification of the training data. Additionally, the interactive direct evaluation process enabled the user to identify problems with a model at a fine level of granularity, as she explored the models’ outputs for specific gestural inputs. This exploration allowed her to judge a model’s quality according her expert knowledge of how it should behave for particular gestures, which in turn allowed her to drive improvements to the model based on precise knowledge of how she wanted the model’s performance to change. Interaction thus offers a more direct way of building models based on users’ priorities, compared to the more coarse-grained strategy for model improvement used in a conventional supervised learning approach, which is to search for algorithms and features that yield higher overall accuracy. Furthermore, the interactive process enabled the user to make adjustments to the problem definition and scope, and to her strategies for providing the most effective training data, based on her evolving experience with the system.

1.6.3 The Wekinator Software

Building the bow gesture classification models engaged the use of nearly all of the Wekinator’s current interfaces and features. The spreadsheet dataset editor was used for deleting segments of training data that were noisy or contained mistakes, and the graphical dataset editor was used extensively for visualizing the sensor features and manually cleaning and re-labeling the training data. Both cross-validation and direct evaluation were useful mechanisms in the interactive model building process: cross-validation allowed an efficient means of comparing algorithms on the same data, and direct evaluation using a variety of visualizations of the model outputs allowed the cellist to explore models’ behavior and evaluate them against her own subjective
criteria. The foot pedal input was critical to allowing an uninterrupted, hands-free means of training data collection and model evaluation.

Several additional features could make the Wekinator a better tool for this sort of gestural recognition task, to allow for more flexible handling of features and support more appropriate modeling algorithms. First, for complex modeling tasks such as articulation, it was not clear a priori which features that could be computed from the bow sensors would be most useful for classification. Based on previous published work finding that minima and maxima of acceleration and velocity were highly relevant features, we integrated these measurements into our feature extractor application. However, similar gestural modeling problems in music—for example, identifying the beat patterns of a conductor—would likely require a different set of features to be computed from the sensor or video input features. (Notably, analysis of human conducting and interfaces for musical control using conducting gestures have been studied extensively, for example by Boie et al. [1989], Lee et al. [1992], and Nakra [2000], among many others.)

Experimentation with different types of features is likely to be necessary for complex modeling problems for which the user is not able to redefine or restructure the problem so that it can be modeled appropriately with the given features and algorithms. The Wekinator itself currently offers no way to experimentally add new features using its GUI, other than those already offered in the set of standard meta-features. Support for interactively designing new features to be computed from the existing feature set, and for retroactively adding them to the current training set, could be very helpful for these types of problems.

Additionally, much of the prior work on bow gesture recognition has applied various segmentation methods to the incoming sensor values, so that a new data example is computed for each note, as in work by Rasamimanana et al. [2005], or each full bow, as in work by Peiper et al. [2003]. The Wekinator does not currently support
any such segmentation; rather, each incoming feature vector is treated as a separate instance to be added to the training set or classified. This can make the learning problem more difficult, in that it can lead to a higher degree of variance among training examples within a given class; for example, the “spiccato” training instances were extracted not only from moments in time when spiccato notes were being played, but also from moments in time where no articulation label truly applied, such as moments in between notes and during changes in bow direction. Segmentation methods are useful in other classification problems beyond gesture analysis, as well; in music information retrieval, for example, a common method of extracting feature vectors from a music signal is to first detect the locations of note onsets, then extract a single feature vector per note [West and Cox 2005]. Therefore, it may be sensible to add support within the Wekinator for specifying a segmentation mechanism that will dynamically identify moments in time when a training or classification instance should be constructed, and accordingly readjust the windows over which meta-features are computed.

Finally, though the current suite of classification algorithms performed well on these gesture classification problems, there are other bow gesture analysis tasks for which other learning algorithms might be more appropriate. For example, the cellist was also interested in building classifiers for user-defined control gestures, such as the discrimination among digits (e.g., “1” through “5”) drawn in the air with the bow. In a brief experimentation with digit classification, we were able to construct a 5-digit classifier that worked reasonably well (rated “8” by the cellist and obtaining 98.9% cross-validation accuracy), but the performance of the model was quite sensitive to the speed of the gesture and the size of the “history” meta-features used. While this type of task can be supported moderately well using the “history” meta-feature, which represents temporal variation directly in the feature vector by concatenating a set of the most recent values of an incoming feature, other learning algorithms that
are structured to take temporal behavior into account might be more appropriate, such as hidden Markov models (Bishop 2007).

Additionally, the Wekinator’s current set of algorithms cannot meaningfully use training examples that have partial or ambiguous class memberships. This type of designation would be appropriate, for example, when switching between up and down bows, or even changing from one articulation to another. A learning algorithm capable of effectively handling these designations could make the training example creation process more natural (e.g., the user would not have to create artificial additional classes for ambiguous examples, as was done in the Bow Direction task above), and it might produce more meaningful posterior distributions (e.g., representing a smoothly changing decision boundary as a gesture moves from one class to another). Therefore, future work might also examine incorporating heuristic methods for handling partial class memberships into the Wekinator’s existing set of algorithms, and/or adding probabilistic learning algorithms that can handle ambiguity in a principled manner.

1.6.4 Further Discussion

We further discuss the implications of this work regarding model evaluation, algorithm and interface design, and the larger role of interaction in supervised learning in Chapter ??.

1.7 Conclusions and Future Work

In this chapter, we have discussed the application of the Wekinator to classification of standard cello bow gestures. Through an interactive machine learning process, a cellist was able to construct working models for eight bowing gestures. The cellist’s subjective rating of these models was quite high—“9” to “10” on a 10-point scale—and
their cross-validation accuracy is comparable to or better than previously published results in bow gesture modeling.

The most significant findings of this work entail a greater understanding of the range of interactions with the training data, algorithms, and features that support building an effective classifier for natural musical gestures; an understanding of the priorities and evaluation criteria of the cellist and information about how the subjective quality or usefulness of a model might not be well-described by a cross-validation accuracy score; and confirmation that an interactive supervised learning process can still lead to and benefit from evolutions in users’ knowledge and goals, even for modeling problems whose structure and goals are fixed and beyond the user’s control.

We have discussed above the future improvements to the Wekinator software that could be helpful in modeling similar problems. Additionally, a wider study with more cellists (or other string players using the K-Bow) would be valuable in gaining a broader perspective on inter-player variations in bowing techniques and in studying other musicians’ criteria for evaluating model correctness, their strategies for effectively creating models, and the ways in which their interactions with the system inform them about machine learning and about their own technique. While some prior work in bow gesture classification has motivated by pedagogical applications, the motivation seems to focus on the use of models trained by experts to give feedback to novice players. In contrast, a new vein of work might also use the process of interactive model building itself to elicit critical reflection from students as they are prompted by the system to consider how to provide a set of gesture examples that successfully demonstrate the essential characteristics of a particular technique.
Chapter 2

Conclusion

2.1 Summary and Contributions

In this thesis, we have examined applied machine learning through the lens of human-computer interaction. In doing so, we have created, studied, and improved systems for users to interactively apply supervised learning algorithms to their work in computer music composition, performance, and instrument design. We have created a useful software tool that has aided students and professional composers in creating new musical works, and in doing so, we have demonstrated the feasibility and efficacy of interactive machine learning in this application domain. Our work with users has led to a clearer characterization of the requirements and goals of interactive machine learning users and of the different roles that interaction may play in allowing them to design and evaluate systems, to learn to become more effective users of machine learning, and to work creatively. As a result, this work has both empowered musicians to create new forms of art and contributed to a broader HCI perspective on machine learning practice.
In this section, we present a summary of our work and highlight the contributions that are most significant to future research in HCI and machine learning, as well as to research and creative work in computer music.

These contributions include:

1. A new software tool allowing real-time human interaction with supervised learning algorithms and, within it, a new “playalong” interaction for training data creation.

2. A demonstration of the important roles that interaction—encompassing both human-computer control and computer-human feedback—can play in the development of supervised learning systems, and a greater understanding of the differences between interactive and conventional machine learning contexts.

3. A better understanding of the requirements and challenges in the analysis and design of algorithms and interfaces for interactive supervised learning in real-time and creative problem domains.

4. A clearer characterization of composers’ goals and priorities for interacting with computers in music composition and instrument design, and a demonstration that interactive supervised learning is useful in supporting composers in their work.

5. A demonstration of the usefulness of interactive supervised learning as a creativity support tool.

2.1.1 The Wekinator

Our work has produced a new software tool, the Wekinator, that allows end users to interactively apply supervised learning learning to their work in real-time problem domains. It is general-purpose in nature, in that users may apply it to creating
trained models that analyze gesture, audio, or other arbitrary real-time input signals and produce outputs that drive sound synthesis, visualizations, or other arbitrary dynamic processes. It is tailored for use in music: it comes packaged with a set of audio and gesture feature extractors and example synthesis patches; and it uses Open Sound Control (OSC: Wright and Freed 1997), a communication protocol common in music and media software, to support compatibility with other software systems designed and used by composers. It supports a rich set of user interactions with the supervised learning process, including the creation of training data by real-time demonstration and the evaluation of trained models through real-time demonstration of testing examples and observation of model behaviors. It also supports interactive, iterative modification of the training data, the selection and configuration of learning algorithms, and the selection of features.

### 2.1.2 Playalong Learning

In addition to enabling users to create training data by demonstrating gestures that correspond to a fixed set of model outputs, we have enabled Wekinator users to create training using a “playalong” interface in which they gesture along with a changing “score” of model outputs, as if they were controlling those outputs in real-time.

Some of the composers who used the Wekinator found playalong useful in creating data that captured more fine-grained aspects of their ideas for how model outputs should vary with the input features. Additionally, composers used playalong to engage their own musical and physical expertise more deeply in the model-building process. It allowed them to use a more embodied approach to instrument design and to create models that captured something about how an instrument should “feel” in expressive, real-time performance.
2.1.3 Demonstrating the Feasibility and Usefulness of Interactive Supervised Learning in Music

We have observed and collaborated with a variety of musical users applying the Wekinator to their work. During a 10-week participatory design process with seven composers, we made many improvements to make the software more usable and useful to composers building expressive new instruments. We used the Wekinator as a teaching tool with 22 undergraduate students, who used it to build new expressive controllers and gesture recognition systems. We also collaborated with a professional cellist/composer using a sensor-equipped bow to interactively build classifiers capable of recognizing standard cello bow gestures. Finally, we conducted interviews with three composers—an undergraduate student, a graduate student in composition, and a professional composer—who have used the Wekinator in publically-performed computer music compositions.

The outcomes of this work with Wekinator users first underscore the feasibility and usefulness of applying interactive supervised learning to a range of applications in computer music. Most of the users studied in this work knew very little or nothing about machine learning before starting their work with the Wekinator, yet the software enabled them to apply standard supervised learning algorithms effectively to create a variety of interactive music systems. These systems included expressive new musical instruments, which used supervised learning models to translate sensed performer gestures into continuous changes in synthesis algorithm parameters; as well as systems for the recognition of standard or novel discrete performance gestures (such as cello bow articulations or physical hits to the laptop) and the translation of these gestures into changes in sound or visualizations.

Students were successful in building accurate classifiers and expressive instruments using the Wekinator in their coursework. The cellist/composer was able to build classifiers with which she was highly satisfied, and which performed comparably to
or better than classifiers created by researchers using conventional (non-interactive) machine learning on the same problems. Composers indicated that the Wekinator enabled them to work more efficiently and to build new instruments that were more musically expressive than those built in other ways. Furthermore, several composers remarked that the Wekinator enabled them to attempt and succeed in creating musical systems or compositions that they would not have attempted or even imagined without the use of the software.

2.1.4 Demonstrating The Usefulness of Interaction in Supervised Learning

Our studies of these users have also led to new knowledge about the roles that interaction can play in allowing end users to successfully create supervised learning systems. Specifically, in our studies, interactive creation of the training set allowed users to define the scope of the learning problem, to communicate the essential characteristics of each class, to preemptively minimize error and cost of the trained model, and to sketch out areas of interest and denote boundaries of the input gesture and output sound spaces. The “playalong” interface for data creation allowed users greater control over the learning concept expressed in the training set and further engaged their physical and musical expertise in the design process. Interactive modification of the training set allowed users to correct errors, to take advantage of new and desirable sounds they discovered during their work, to make models more complex, and to rollback changes they had made. Interactive evaluation of trained models enabled users to evaluate models against their goals for the learning process; assess how they might improve models through modifications to the training data and learning concept; compare models produced by different learning algorithms; and identify problems with the training data not identifiable using cross-validation. Interactive evaluation also trained users to become better machine learning practitioners, served as a method
of “practice” in which they assessed and improved their own abilities to behave in ways that produced the desired model outputs, and led to refinements and changes in users’ goals for the machine learning system.

Through repeated iterations of model evaluation and modification, users were able to construct interactive systems that met their goals. Significantly, throughout these iterations, users worked to improve the performance of the trained models according to their criteria for success, while also refining their goals for how the trained models should function and how the models would ultimately be used. Users sometimes simplified the problem when models performed poorly, sometimes increased the complexity of the problem when models performed well, and sometimes changed the nature of their goals entirely when they discovered unexpected new behaviors in the models. The abilities to iteratively evaluate models against an array of subjective criteria and to change the training set to reflect their evolving goals for the system were crucial.

2.1.5 Understanding the Differences between Interactive and Conventional Machine Learning

These studies also led to a better understanding of differences between interactive and conventional supervised learning that may be useful in designing new learning algorithms for interactive applications. Importantly, generalization accuracy was sometimes relevant to users’ goals for the system, but it was not the only or most important evaluation criterion that users employed. This suggests that future work might investigate whether existing or new algorithms that build models according to other criteria might be appropriate for interactive scenarios, and it suggests that evaluation by human users should supplement conventional metrics such as cross-validation accuracy in the evaluation of models intended to be used in interactive contexts.
Additionally, the size of the training sets employed by users in this work was often small compared to those used conventional supervised learning. This offers a practical advantage, in that short training times lead to less interruption between model modification and model evaluation actions. We hypothesize that the smaller size of the training sets is due to users adjusting the difficulty of the target learning concept to ensure that it is possible to learn with a training set size that is both feasible to interactively create and allows short training times, and/or to the fact that users learn through experience to provide training examples that are more effective than randomly-chosen examples in modeling the target concept.

2.1.6 Understanding Requirements and Challenges of Algorithm and Interface Analysis and Design

Through our analysis of how users interacted with supervised learning within the Wekinator, the characteristics of the software they most valued, and the challenges to effective interaction that remained following our work with composers to improve the system, we have come to understand how the interactional affordances of a learning algorithm play a role in shaping its usefulness and usability. Among the interactional affordances that were key to making standard supervised learning algorithms usable in this work were their low training time, their capability for building models for the chosen learning concepts using a small number of training examples, their fast running time, and their ability to be “steered” in different directions via users’ modifications to the training set. Future work to improve algorithms for interactive contexts could focus on adding new interactive affordances, for example mechanisms similar to active learning that prompt the user to provide training examples that are anticipated to be useful. Or, future work could focus on making the algorithms more amenable to embedding in a usable user interface, for example by designing algorithms with
tunable parameters that can more be easily manipulated to change the model along dimensions of interest to a user.

Characteristics of the interface that were key to making the Wekinator usable included exposing the above affordances of the learning algorithms, rather than hiding them from the user (for example, conventional machine learning interfaces hide the affordance for tuning model performance using the training set); enabling effective control over feature selection to allow users to create systems with independencies between features and outputs, and enabling training example selection to allow users to employ a training data creation strategy in which they considered one output parameter at a time; and providing feedback to the user about the state of the learning system and the efficacy of users’ actions. Future work might improve the user interface to more effectively address the challenge of providing useful, timely, and fine-grained feedback over learning system state, the actions available to a user to interact with the system, the consequences of past actions, and the likely consequences of future actions. In particular, this work might explicitly address the challenge of providing this information in a manner that is suitable to machine learning novices.

The ways that users employed algorithms and interfaces, and by extension, the properties of the algorithms and interfaces that were important to them, were contingent on several characteristics of the problem domain. In the applications studied in this thesis, users were domain experts (i.e., experts at performing the gestures used to provide input to the models). The users were building models for themselves or for other users they understood well, creating models that would be used in a real-time context, and working with a set of goals that ranged from mostly inflexible to completely flexible. Future work might investigate interactive supervised learning in other domains whose characteristics vary along these dimensions to identify commonalities or differences in users’ interaction and algorithm needs, thereby generalizing
our findings to other domains and identifying the most fruitful avenues for improving algorithms and interfaces for a wide population of users.

2.1.7 Demonstrating the Usefulness of Interactive Supervised Learning as a Creativity Support Tool

Users’ work with the Wekinator has demonstrated that interactive supervised learning can be a useful tool in creative contexts: the Wekinator enabled users to not only work more efficiently and productively than other tools they had used, but it also inspired them and helped them to create musical compositions and systems to express themselves in new ways. In fact, many of the ways that users in our studies employed the Wekinator can be understood as taking advantage of the capabilities of interactive supervised learning to function as a creativity support tool. We have discussed how interactive supervised learning effectively addresses many of the requirements of creativity support tools as proposed in work by [Resnick et al. (2005)](#) and others. In particular, our work has demonstrated that interactive supervised learning can effectively support creative work through assisting in the exploration of design possibilities, rapid prototyping, and sketching; accommodating novice and expert users working on a range of tasks; supporting a high-level and holistic approach to design; integrating well with other software tools; enabling learning algorithms to be treated as black boxes when appropriate; and providing users with access to surprise, complexity, and inspiration.

Furthermore, Wekinator users felt strongly that they benefited from the physical, embodied approach to design that was supported by the software. Through engaging embodied interactions with the training data and trained models, interactive supervised learning can leverage embodied knowledge, skill, learning, and memory in the creative design of interactive systems.
2.1.8 Understanding Composers’ Priorities for Human-Computer Interaction in Composition

Finally, our work has led to a clearer characterization of composers’ priorities for human-computer interaction in composition and instrument design. Composers and students creating new instruments valued the speed and ease with which they could create and explore mappings, approaches to design that privileged the gesture-sound relationship via physicality and abstraction, access to surprise and discovery, access to complexity, the ability to balance surprise and discovery with predictability and control, and tools that presented an invitation to play. Additionally, the cellist valued the speed and ease with which she could construct accurate and musically-useful models of her natural cello gestures.

Composers using the Wekinator to build new, expressive instruments especially valued the way they could use the Wekinator for inspiration, surprise, and discovery. Their work with the Wekinator was not a one-way exercising of control over technology to build a model that met their specifications; rather, it was a mutually influential, richly interactive process. By considering the Wekinator as a meta-instrument, these types of interactions can be seen as fulfilling the requirements of real-time musical interactivity put forth by composers such as Moon (1997) and Chadabe (2002) in their writings about performance-time interactions.

2.2 Future Work

2.2.1 Improvements to the Wekinator

Our discussions with composers and musicians have provided us with a rich set of ideas for further improving the Wekinator as a creativity support tool and meta-instrument. Some of these improvements involve supporting new interactions to benefit composers’
work. For example, adding parameter smoothing to the playalong process will enable
composers to create training datasets capable of faithfully representing a broader
range of goals, and to create training data in a way that better takes advantage of
composers’ embodied expertise. Also, adding mechanisms to visualize the training
dataset and decision boundaries might provide additional means to understand how
the learning algorithm is working, thereby helping machine learning novices to better
understand the effects of their actions and helping users to identify and fix problems
when the models do not work as they expect.

Another set of improvements will focus on making the software interface even
simpler and easier for more people to use. The current terminology in the interface
(especially with regard to features and parameters) is confusing to novice users and
should be changed or augmented with more helpful contextual information. Fur-
ther integration with OSC—for example, enabling users to send OSC commands to
load and save learning systems—would enable users greater freedom to embed the
Wekinator within their own systems.

Throughout these changes, we plan to maintain the general-purpose nature of the
Wekinator so that it remains possible to apply the system to non-musical problems.
We would like to work with non-musical users, for example users designing interactive
art or gaming systems, to better understand their needs and ensure that the Wekina-
tor supports this work well. We would also like to apply the Wekinator to real-time
audio analysis problems in computer music performance, an initial goal of our work
that has not yet been realized.

An additional set of improvements will focus on continuing to improve the use-
fulness of the Wekinator as a tool for conducting research in interactive supervised
learning in real-time domains. We will strive to make it as easy as possible for our-
selves and other researchers to add new learning algorithms, new modes of feedback
to users, and new mechanisms for logging and understanding users’ actions with the system.

2.2.2 Research Directions in HCI, Machine Learning, and Computer Music

This work has helped to formulate several research directions surrounding the improvement of software systems for end-user interactive machine learning and the application of interactive machine learning to new problems and domains. First, our work has shown that users applying interactive machine learning may or may not be concerned with the generalization accuracy of models, and even when generalization accuracy is important, it may be only one of several characteristics that users care about. Through future work investigating what characteristics of trained models may be important to users across different application domains, we might arrive at a better understanding of how to design algorithms that create models with these characteristics and how to design interfaces that allow users to evaluate models against these characteristics. Furthermore, future work should investigate whether learning algorithms capable of prioritizing training accuracy at the expense of generalization accuracy may be better-suited to interactive applications in which users exercise control over the learning problem primarily through modifications to the training set.

Other future work might combine HCI and machine learning approaches in other ways. For example, research into the design of algorithms with parameters whose effects are clearly understandable to users and likely to affect dimensions of a model that users care about could result in more usable algorithms. Alternatively, work that combines theoretical understanding of algorithms and novel user interface techniques might better enable users to employ the parameters of standard algorithms to more effectively improve models against their criteria for success.
Our findings suggest that the degree to which users’ goals for supervised learning are flexible impacts the ways in which they may use interaction to build models to achieve their goals. Future work might study users completing different types of tasks in a more controlled environment to better understand how users’ interface and interaction requirements differ according to the degree of this flexibility. Additionally, future work might consider other problem domains in which users’ goals are relatively fixed—including domains where conventional machine learning is often applied—and investigate the extent to which interactive approaches can still exploit the aspects of the learning problem (e.g., number and nature of classes, sources of features) that remain flexible.

Another research direction concerns the creation and analysis of additional interface mechanisms to provide richer and more useful feedback to users, including users who are machine learning novices. In our observations, users employed interactive evaluation of models for a variety of purposes, including to assess the quality of trained models against their evaluation criteria, to determine which actions they might take to improve those models, to learn what might or might not be possible to accomplish using the given algorithms, and to learn which of their actions were most effective in creating the types of models they wanted. These types of feedback were all important to users, and future work might investigate new user interfaces to provide this feedback more efficiently and effectively, to provide complementary information, or to provide information tailored to novice users who are just learning how to use interactive machine learning effectively.

Our research has highlighted the ability for interactive machine learning to effectively support creativity and an embodied approach to building interactive systems. We are excited about the prospects to explore how users in other creative and interactive problem domains, including interactive art and game design, might benefit from interactive supervised learning tools. Additionally, we observed the capability
for interactive supervised learning to cause users to reflect on their own abilities and actions; for example, users sometimes discovered through their failure to build an accurate classifier that the training data they created was not consistent or clear. Future work could therefore investigate the potential of interactive supervised learning in pedagogical or therapy applications where users may benefit from mechanisms for learning about the consistency of their actions.

Finally, further work with composers, instrument designers, musicians, and music students will likely illuminate new avenues for interactive supervised learning to be applied more effectively and more broadly to creative work in computer music. New algorithms could be designed to incorporate behaviors and tunable parameters tailored specifically for application to the design of new musical instruments. New interfaces could also be built to allow new types of supervised learning systems. For example, systems with hierarchical relationships between models, in which the outputs of some models serve as input features to other models, could be useful for problems in which more sophisticated segmentation mechanisms are necessary, as well as for allowing synthesis or other music-producing processes to be controlled in new ways. Or, interfaces and algorithms could be constructed that enable the trained models to smoothly evolve or change over time in response to human actions or random processes. In any case, we hope to continue to engage composers and musicians in participatory design of these systems, as we have found our work with users to be crucial in improving and evaluating our software, understanding their priorities for incorporating technology into their work, and identifying key research questions pertaining to interactive supervised learning in music and other domains.
2.3 Conclusion

Our work has emphasized the importance of considering the human context of machine learning practice. By providing users with the ability to engage in a richer set of interactions with supervised learning algorithms, the software we have created has enabled people to put standard learning algorithms to use to do their work more efficiently and effectively, and to imagine and accomplish goals that were not possible with the tools previously available to them. Our work has shown that these interactions can empower end users to build interactive systems for themselves and for others while employing an approach to design that leverages their domain knowledge, embodied expertise, and understanding of the context in which the trained models will be used. Furthermore, these interactions can inspire users to consider new design possibilities and engage in self-reflection and growth as machine learning users and creative beings.

In our work, the principles and methodologies of HCI research, the computational possibilities presented by machine learning, and the knowledge, values, and ideas of domain experts have been instrumental in shaping our research questions about how the computational and interactive affordances of learning algorithms can be exploited to meet real-world application requirements and support users’ values. Having demonstrated the feasibility and usefulness of applying interactive supervised learning to the musical problems studied in this thesis, we are excited by the opportunities for future work that, by continuing to build on the intersection of HCI, machine learning, and application domain knowledge, will lead to new algorithms, interfaces, and interactions capable of supporting even more users in applying supervised learning more effectively and to more problems.
Bibliography


