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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>1 Teaching Interactive Systems-Building with the Wekinator</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Background, Motivation, and Goals</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 The Princeton Laptop Orchestra</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2 Supporting and Studying Interactive Systems-Building in PLOrk</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Method</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Observations</td>
<td>9</td>
</tr>
<tr>
<td>1.4.1 Interactive Systems Built</td>
<td>9</td>
</tr>
<tr>
<td>1.4.2 Actions Performed</td>
<td>10</td>
</tr>
<tr>
<td>1.4.3 Interaction Over Time</td>
<td>13</td>
</tr>
<tr>
<td>1.4.4 Model-Building Strategy</td>
<td>16</td>
</tr>
<tr>
<td>1.4.5 Human Learning and Adaptation</td>
<td>20</td>
</tr>
<tr>
<td>1.4.6 Success in Using the Wekinator</td>
<td>23</td>
</tr>
<tr>
<td>1.4.7 Effects of Task and Order</td>
<td>24</td>
</tr>
<tr>
<td>1.5 Discussion</td>
<td>25</td>
</tr>
<tr>
<td>1.5.1 The Wekinator as a Teaching Tool</td>
<td>25</td>
</tr>
<tr>
<td>1.5.2 Interactive Supervised Learning and Task Type</td>
<td>28</td>
</tr>
</tbody>
</table>
1.5.3 Improving the Wekinator .......................................................... 30
1.5.4 Further Discussion .................................................................. 31
1.6 Conclusions and Future Work ...................................................... 31

Bibliography .......... 33
List of Tables
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The PLOrk Ensemble, 2006</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>A PLOrk Station</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Gestural Inputs and Synthesis Algorithms Used</td>
<td>10</td>
</tr>
<tr>
<td>1.4</td>
<td>Students’ Success in Building Models</td>
<td>11</td>
</tr>
<tr>
<td>1.5</td>
<td>Number of Times Student Retrained Models</td>
<td>12</td>
</tr>
<tr>
<td>1.6</td>
<td>Actions Taken to Improve Models</td>
<td>12</td>
</tr>
<tr>
<td>1.7</td>
<td>Time to Complete Assignment</td>
<td>14</td>
</tr>
<tr>
<td>1.8</td>
<td>Time Breakdown</td>
<td>15</td>
</tr>
<tr>
<td>1.9</td>
<td>A Wekinator Re-Imagining of Gershwin’s “Summertime”</td>
<td>26</td>
</tr>
<tr>
<td>1.10</td>
<td>A Wekinator “Instrument Petting Zoo” Final Course Project</td>
<td>27</td>
</tr>
</tbody>
</table>
Chapter 1

Teaching Interactive Systems-Building with the Wekinator

1.1 Introduction

In this chapter, we discuss our study of the Wekinator as a tool for teaching interactive systems-building to undergraduate students enrolled in the Princeton Laptop Orchestra course. We focus on students’ use of the Wekinator in completing a structured systems-building assignment, as well as a longitudinal analysis of students’ work with the Wekinator over the course of the semester. Through logs of students’ interactions with the software and their written feedback, as well as the final outcomes of their coursework, we examine how they used interactive learning to create musically expressive and accurate models, the ways that the Wekinator influenced their work, and the efficacy of the Wekinator as a teaching tool.

We begin this chapter by providing background on the Princeton Laptop Orchestra course. We discuss how students’ work with the Wekinator fit into the goals of the
course, and we present our research goals for our study of its use in this context. We
describe the method used for studying students during their use of the Wekinator in
the midterm assignment, and we present our observations regarding the interactive
systems the students built, the actions they performed, how they used their time
during the assignment, their strategies for model building, the ways that they learned
and adapted, their degree of success in the assignment, and the effects of task type
and order on students’ behaviors. Drawing on our observations and other experiences
in the course, we discuss the usefulness of the Wekinator as a teaching tool, the
relevance of task type in studying and supporting interactive supervised learning,
and improvements to the Wekinator suggested by this work. We later draw on this
work in Chapters ?? and ??, in discussing algorithm and interface design in interactive
supervised learning, the use of interactive machine learning by novices and in creative
work, and the larger role of human interaction in applied supervised learning.
1.2 Background, Motivation, and Goals

1.2.1 The Princeton Laptop Orchestra

The Princeton Laptop Orchestra (PLOrk), pictured in Figure 1.1, is an undergraduate teaching initiative and performance ensemble, created in 2005 by Princeton faculty members Dan Trueman and Perry Cook (Trueman et al. 2006). Drawing inspiration from historic electronic performance ensembles such as the League of Automatic Composers and the Hub (Bischoff et al. 1978; Brown and Bischoff 2002), as well as from Trueman’s own work creating digital meta-instruments such as the BoSSA (Trueman and Cook 2000), a motivation for forming PLOrk was to experiment with making music with a larger group of performers playing laptop-based meta-instruments (Trueman 2007). Another inspiration for PLOrk was work by Cook and Trueman (1998), Wessel (1991), and Trueman et al. (2000), exploring how to design new electronic instruments with a physical presence more like that of acoustic instruments, specifically through the use of spherical speakers. In PLOrk, many human laptop performers share the stage, each using their own laptop and hemispherical speaker, shown in Figure 1.2.

At its inception, the PLOrk ensemble consisted of fifteen human laptop players. Through the years since 2006, the size of the ensemble has grown to over 25 performers, and the ensemble has performed both full-ensemble and “chamber” works (for fewer performers) composed by Princeton faculty, students, and alumni, as well as guest composers. Notable past collaborators include renowned composers Paul Lansky and Pauline Oliveros, tabla virtuoso Zakir Hussain, and experimental electronica duo Matmos.

In addition to being an active performing ensemble, the Princeton Laptop Orchestra is an undergraduate educational initiative. PLOrk was first taught as a course at Princeton University in Autumn 2005 as a Freshman Seminar. It has been taught
every year since then as an undergraduate course in Computer Science and Music, open to students of any major and any academic year.

Since 2008, the PLOrk class has included assignments and weekly lectures devoted to exposing students to a variety of theoretical and practical topics in computer music composition and performance\textsuperscript{1}. The class introduces student to topics including object-oriented design and programming (primarily using the ChucK language \cite{Wang and Cook 2003}), software engineering, signal processing, audio synthesis, and interactive systems-building. A goal of the course is to give students from non-technical majors exposure to these areas of computer science and engineering, and to give students in technical majors practice in applying their expertise to new domains and in creative ways.

1.2.2 Supporting and Studying Interactive Systems-Building in PLOrk

In Spring 2010, the PLOrk course emphasized interactive systems-building. In many assignments, as well as in the midterm and final projects, students built new musical interfaces that could be used in performance and as interactive installations. These interfaces ranged from very simple, for example using the laptop’s internal accelerometers to control the pitch of a synthesis algorithm in an assignment, to quite elaborate, for example using a custom vision-based motion tracking program to sonify the walking patterns of visitors to the music building in a final project installation. Interactive systems-building projects were used both as a tool to help students learn course concepts (e.g., software engineering and signal processing), and as an end in themselves, to empower students to create interactions they found to be fun, compelling, and expressive.

One of the topics emphasized in the course was the “mapping problem,” described in Section ???. Students were first taught about “explicit mappings,” in which gestural controllers are built by explicitly programming the ways that gestural controller or sensor values drive the parameters of a synthesis algorithm. Students built several explicit mappings in course assignments by using SMELT repository code (Fiebrink et al., 2007) to extract gestural input signals from USB HID devices (i.e. standard input and gaming devices; see Section ???) and native laptop inputs, then writing their own ChucK code to change sound synthesis parameters based on the values of these inputs.

Students were also taught about “generative” mappings that construct the gesture-to-sound mapping function from a set of training examples, and they were given a high-level introduction to how machine learning algorithms can be used to generate mappings from training data. Prior to the Wekinator, there existed no appropriate tools to allow students (especially those with limited programming
ability and machine learning knowledge) to create generative mappings for using arbitrary input devices (including HID devices, audio and video inputs, and other native laptop inputs) to control arbitrary synthesis programs. The Wekinator is therefore an attractive tool for enabling students to quickly and easily build working interactive systems and to learn about generative mappings and machine learning in a hands-on way.

The goal of the work described here was to study the first use of the Wekinator as a tool for teaching interactive systems-building in an undergraduate classroom. The Wekinator was used in the PLOrk course to teach students about the breadth of ways that computers can be used in interactive performance, to enable students to build interactive systems that they enjoyed and found musically useful, to teach them about the tradeoffs between explicit and generative mapping strategies, and to foster the development of their object-oriented design, signal processing, and music programming skills as they designed feature extractors and sound synthesis programs to be used in conjunction with the Wekinator. In our study of students in the course, we hoped to discover how the Wekinator was useful in meeting these goals, and to discover what about it could be improved for future pedagogical applications. Additionally, we aimed to learn about how students used the Wekinator in building interactive instruments and installations, including the actions they performed, the strategy they employed in building models, and how the Wekinator both supported and influenced their creative work.

1.3 Method

While we used the Wekinator throughout the latter half of the Spring 2010 PLOrk course, we focus our study here on our observations of and feedback from the 22 students completing the midterm assignment, which was their first and most structured
assignment using the Wekinator. The midterm assignment served as a hands-on introduction to the Wekinator, and it prepared students to incorporate generative mappings into their compositions in the upcoming midterm performance. Prior to the assignment, students had five weeks of instruction in the ChucK programming language, and they had completed assignments in which they designed explicit mappings using gestures to control sound synthesis algorithms. In course lectures leading up to the assignment, students were introduced to generative mappings and to the standard machine learning algorithms employed by the Wekinator, and they saw demonstrations of the Wekinator performed by the author and by other instructors.

In the midterm assignment, students built two gestural musical controllers using the Wekinator: a continuous controller using neural networks, and a discrete controller using a classifier. Before beginning the assignment itself, the students followed a step-by-step tutorial walking them through how to build two simple instruments for controlling a synthesizer’s pitch using the laptop’s tilt. One of these instruments used a classifier for discrete pitch control, and the other used a neural network for continuous pitch control. Students worked individually on the assignment, and they had twelve days to complete it after it was assigned. As the goal of the assignment was to familiarize students with the Wekinator, students were informed that they would be graded on completing the assignment in a through and thoughtful manner, not based on the musicality of their models or the “correct” use of the Wekinator. They were encouraged to contact the course instructors for help and questions with the assignment.

The midterm assignment was broken into two parts, Part A and Part B, and the order in which students completed these parts was balanced across students. In Part A, students were asked to use the Wekinator’s multilayer perceptron neural networks to build a new, continuous musical control interface that they thought was musically expressive. Students could pick among three pre-built synthesis algorithms, each of
which had between three and nine parameters that affected the sound in non-linear and interdependent ways: a physical model of a bowed string instrument (Smith 1986; Cook and Scavone 1999), a physical model of a hybrid flute/electric guitar called the “blotar” (Stiefel et al. 2004), or an FM synthesis algorithm (Chowning 1973). Students could also pick among four gestural control methods for which the Wekinator provided built-in feature extractors: a GameTrak USB tether HID device (see Section ?? and Figure ??), a Logitech joystick HID device, the laptop’s internal accelerometers, or the webcam color tracking feature extractor described in Section ??.

After students had completed the construction of the musically expressive controller, they were asked to discuss their work in a series of five short-answer questions. Specifically, they were asked to explain their choice of gestural controller and synthesis algorithm, discuss their goals for how the instrument would be used to control sound through gesture, and describe their strategy for building an expressive model. Additionally, they were asked to rate on a 5-point Likert scale whether they were successful in building an expressive model, building a model whose gesture-to-sound mapping could be controlled in a predictable manner, and getting the Wekinator to learn what they wanted it to. Students were also invited to share any other comments on what they learned, what they found confusing, or anything else.

In Part B of the assignment, students were asked to use a classification algorithm to build a model performed reliable gesture classifications into any gesture categories of their choosing. The class labels output by model were used to control a simple, pre-built synthesis algorithm driven by a single discrete parameter. Students could choose either a melodic synthesizer whose parameter controlled pitch, or a drum machine whose parameter controlled the number of drum loops or layers to play simultaneously. Students could also choose among the same four gestural controllers as Part A: the tether, joystick, laptop accelerometers, or webcam color tracker.
After Part B, students completed five short-answer questions about their work building the reliable gesture classifier. These questions were identical to those answered after Part A, with the exception that they were asked about their success in building a model that provided reliable gesture classifications, rather than building a model that was musically expressive.

Students were provided with a special version of the Wekinator software that logged their actions as they completed Parts A and B. All actions such as training models, recording data, evaluating and running models, and changing algorithms and features were recorded to a file in real-time. The students emailed us the log files at the completion of the assignment. Students were aware that the logging was being performed in order to discover how they were using the Wekinator, and that the log contents would not be used in determining their grades.

All students completed the assignment, and all but one student provided the Wekinator log files. Eleven of those students completed Part A first and ten completed Part B first.

1.4 Observations

1.4.1 Interactive Systems Built

In the midterm assignment, all students successfully completed Part A and Part B, constructing an expressive continuous controller and a reliable gesture classifier. (Because one student was unsuccessful in retrieving the Wekinator log files, all logging analyses are performed over only 21 students.) Nearly all students used the synthesis algorithms and gestural control inputs specified in the assignment instructions. Figure 1.3 shows the number of students who chose each input and synthesis algorithm for Part A and Part B.
Figure 1.3: The number of students who chose each gestural input and synthesis algorithm, for Part A and Part B.

Figure 1.3 shows students’ levels of agreement with several statements regarding the success with which they were able to use the Wekinator in the assignment, rated on a 5-point Likert scale. In general, students were very successful in using the Wekinator to accomplish the goals of the assignment; the average agreement with all statements was above 4.0. Statement S1, “My model is musically expressive,” received the lowest average score overall (4.1), and several students indicated in the written component of the assignment that they felt limited by the expressive potential of the synthesis algorithm that they chose, not by the Wekinator or their mappings.

### 1.4.2 Actions Performed

In the midterm assignment, the logging data and students’ written work clearly indicate that students employed an iterative approach to interactive model building, in which they retrained the model multiple times following changes to the training data or algorithm. Figure 1.3 shows the number of times a model was retrained over the course of each task, by each student (not counting the first time the model was trained). More than half of the 21 students retrained the model at least once, but there was great variation in how many times students retrained the models, up to a maximum of 17 times by one student in Part B. The average number of times a
(a) Statement S1: “My model is musically expressive.” (Part A only)

(b) S2: “My model provides reliable gesture classifications.” (Part B only)

(c) S3: “I can reliably predict what sound my model will make for a given input gesture.”

(d) S4: “Wekinator eventually learned what I wanted it to.”

(e) Average and standard deviation of ratings for statements S1–S4 above.

Figure 1.4: Students’ success against assignment criteria, self-rated as agreement with four statements on a 5-point Likert scale.
Figure 1.5: The number of students who retrained the model a given number of times, during Part A and Part B, not counting the first model training.

model was retrained in Part A was 3.3 (median = 1.0, $\sigma = 4.2$); the average number of times a model was retrained in Part B was 4.9 (median = 1.0, $\sigma = 5.9$).

Between model trainings, students took one or more actions to modify the model, including modifying the training dataset and changing the learning algorithm or its parameters. In Part A, the model was modified by changing the dataset an average of 3.1 times (median = 1.0, $\sigma = 4.2$); the learning algorithm was never changed. In Part B, the model was modified by changing the dataset an average of 4.3 times (median = 1.0, $\sigma = 5.2$), and by changing the learning algorithm or its parameters an average of 0.2 times (median = 0.0, $\sigma = 0.8$). The features used by the models were never changed in Part A or B, though one student changed the controller that she used in the middle of Part B.

Figure 1.6 shows how many times each particular action was taken in order to modify and evaluate a model, for Part A and Part B. Edits made to the training data and algorithms before the first model training are not included, and actions that were performed multiple times between consecutive model trainings (e.g., recording new training examples twice before retraining the model) are counted only once. Each student attempted to improve the model by modifying the training data significantly more frequently than by modifying the learning algorithm or its parameters ($p < .0001$)
Students created training data both using “play-along” recording (in which students set up a parameter “score” and recorded training data by gesturing along to the synthesizer playing this score; see Section ??) and by entering each set of parameter values in the GUI and demonstrating the gesture that corresponded to those values. While only 8 of the 21 students ever used play-along recording, play-along recording was used in 114 of the 188 total occurrences of students recording new training data during the assignment. Students never used the graphical data editor, perhaps because it was not demonstrated to them in class. Four students employed the spreadsheet editor to manually change feature or parameter values; three of these students made between one and three manual edits each, but one student made 95 manual edits as he attempted to fix his models' misclassifications in Part B.
1.4.3 Interaction Over Time

Figure 1.7 shows the time taken by each student to complete each part of the assignment. All but two students took under 45 minutes to complete Part A, and all but one student took under 45 minutes to complete Part B. The average completion time was 27.1 minutes (median = 16.7, $\sigma = 30.5$) in Part A and 16.1 minutes (median = 14.6, $\sigma = 14.3$) in Part B. For most students, the Wekinator offered a reasonably fast way to build a reliable gesture classifier or an expressive continuous controller. Incidentally, the student who took the most time on a task, 145.1 minutes in Part A, did not appear to have had any particular trouble with the assignment; on the contrary, his written response (the longest of any student) indicates that he spent a lot of time experimenting with his blotar controller and exploring several different mapping strategies over the course of the task.

Figure 1.8 shows the average total time spent performing each action with the Wekinator, within Parts A and B. The action to which students devoted the most time was evaluating the model by running it, followed by recording new training data. The time spent to train the models and to compute cross-validation and training accuracy was negligible. In the time spent on “Other” actions, the user was setting
Figure 1.8: The average number of seconds spent performing each action over the course of Part A or B, shown as a proportion of the average total task time.

The time needed to train the models was quite short: 11.5 seconds per training on average in A (median = 4.5, $\sigma = 19.4$) and 0.2 seconds on average in B (median = 0.1, $\sigma = 0.3$). A short training time—much shorter than the minutes or hours that are acceptable in many non-interactive supervised learning scenarios—is important in this context, so that training does not interrupt a user’s interaction with the system and training time does not provide a disincentive to attempt to fix a model if the user
is not satisfied with its behavior. The relatively fast training time can be attributed to the fact that training sets were generally quite small, compared to those used in other applied machine learning problems (see Section ??). The average training set size was 790.1 examples (median = 565.0, σ = 604.7) in Part A and 548.0 examples (median = 424.0, σ = 426.8) in Part B. While the difference in each students’ average training set size in Part A and Part B was not statistically significant (p = .082 using paired t-test), the difference in average training time was significant (p < .05). The difference in training time is due to both differences in learning algorithms and the number of synthesis parameters used in the two parts of the assignment: Part B required the training of only one classifier, while Part A required the training of between three and nine neural networks.

1.4.4 Model-Building Strategy

Both the logging data discussed above and students’ own written remarks on their strategies for model building emphasize an iterative process: students repeatedly built a model from the training data, evaluated it to assess whether they liked it, and attempted to improve it. In this section, we discuss in more detail the actions that students took and the criteria they used to evaluate models, and how they acted on the knowledge gained from model evaluation to attempt to improve the system.

Training Data Creation

A few students began Part A with quite clear ideas regarding the types of gestures and mappings that they wanted the model to learn. The students with a clear plan for how they wanted the instrument to work developed strategies for creating training examples that often focused on just one aspect of the mapping function at a time. For example, one student described his strategy: “I started by first determining which of the HID input parameters corresponded to which axis on the actual tether (i.e.
HID_6 …reflected the length of the right tether string). I then adjusted the neural
networks accordingly so they would only be looking at what I wanted it to …I then
attempted to train each parameter independent of the others by unchecking their
boxes and giving Wekinator some training samples and then training and running it.
Based off how it did here, I went back and gave it more samples before moving onto
the next parameter. After all three parameters were trained I went back and gave
it more samples again for each (again, each parameter was trained independently of
the others) before I was satisfied with how well it could predict what I wanted the
parameters to be for a given motion.”

Many students who did not have such clear a priori goals described a model-
building strategy for Part A that was quite similar to the strategy employed by
composers in Chapter ??: they created training examples matching sound parameters
that they liked with a few different gestures, then let the model “fill in the blanks.”
One student wrote, “I found it the easiest and clearest to create a model from the
outside in, sort of: specifically, I would create the extreme sounds and positions, train
those extremes, and then fill in more information about the transition between the
already learned extremes.” Other students also indicated that they used iterative
retraining to make a model gradually more complex, for example: “I started with
just experimenting with one of the parameters at a time, and slowly worked my way
up so that eventually I was manipulating all three parameters.”

In Part B, some students worked methodically to build the gesture classifier, for
example carefully controlling the way in which parameters and gestures co-varied to
clearly represent their ideas for how the classifier should work. As in part A, some
students used iterative retraining to gradually increase the complexity of their model,
adding new gestures and classes once a model was performing well. For example: “I
didn’t immediately do all of these gestures at first but started with only a few to
get warmed up.” Other students, though, provided training data for all gestures and classes all at once, or did not seem to have much of a conscious strategy.

Just as in Part A, there was a wide variety of among the extent to which students had clear plans about the sorts of models they wanted the Wekinator to build. In some cases, students’ plans for which gestures they would use were informed by which gestures they thought the Wekinator would be able to easily classify. One student who was using color tracking to control the drum machine also formed her plan for the model based on how frequently she expected to use different classes in performance, writing: “I tried to assess which beats I would use more often and correlate them with [feature values] that were easier to obtain on the colour tracker.”

In both Part A and Part B, students’ strategies for creating and modifying the training data evolved as they worked with the Wekinator; we discuss this phenomenon further in Section 1.4.5.

Model Evaluation

The Wekinator allows users to evaluate trained models using two approaches: computing standard metrics (training and cross-validation accuracy) to assess the capability of the algorithm to accurately model the training set, and directly evaluating the trained model by running it on new gestural inputs in real-time and observing its behavior. In order to assess a newly trained model’s performance and determine how to improve it, students employed cross-validation accuracy 13.8% of the time, training accuracy 28.9% of the time, and direct evaluation 93.7% of the time.

Over the course of the assignment, 13 students computed cross-validation accuracy at least once, and 12 students computed training accuracy at least once. Each student computed cross-validation accuracy an average of 0.9 times (median = 1.0, $\sigma = 1.0$) in A, and an average of 1.0 times (median = 0.0, $\sigma = 1.9$) in B. Each student computed training accuracy an average of 1.7 times (median = 0.0, $\sigma = 2.9$) in A
and an average of 1.6 times (median = 0.0, \(\sigma = 3.7\)) in B. Students employed direct, hands-on evaluation significantly more often than they computed cross-validation or training accuracy, as discussed in Section 1.4.2: each student directly evaluated models an average of 4.2 times (median = 2.0, \(\sigma = 4.2\)) in A and an average of 5.3 times (median = 2.0, \(\sigma = 5.3\)) in B.

Students used accuracy metrics and direct evaluation to gain different types of knowledge about the models. In their written work, several students implied that they treated a high cross-validation or training accuracy as reliable evidence that a model was performing well, and students often reported cross-validation accuracy or training accuracy scores of their final models as evidence to the grader that a model was in fact performing well or poorly. At least one student used cross-validation to validate his own model-building ability, writing “Following [dataset creation], I would usually quickly check the cross-validation and training accuracy and see if the Wekinator thought that my model was a good one. If it was, my next step was usually to run the model myself and observe how it reacted to different gestures.” This echoes the findings of Amershi et al. (2010), who observed that users felt pressure to optimize cross-validation accuracy as an end in itself, rather than using it as an informative tool. Six students never computed cross-validation or training accuracy, and some students indicated that they did not understand the metrics or that they found them irrelevant or unhelpful: “. . . I preferred to just run the machine and see how it worked when I was actually trying to use it in practice.”

Students reported using direct evaluation to assess models against a variety of criteria, encompassing correctness, musical expressiveness, complexity, and naturalness. In both Part A and Part B, students identified a model’s behavior as incorrect when it produced an output contrary to what they believed was appropriate and expected. Students also used direct evaluation to assess the model against the assignment goals of musical expressiveness in Part A and reliably accurate classification in Part B.
Like the composers in Chapter ??, students sometimes indicated that, in Part A, complexity and unexpected behavior were in fact desirable properties in a model. One student wrote: “I actually found myself happiest with my model when I had introduced less predictability into it,” and indicated that his goal ended up being to build a model with “an optimal balance between controllability and versatility.” In Part A and Part B, seven students mentioned that it was important to them that, when using the models, the gestures and gesture-sound relationships felt “intuitive” or “natural.”

Model Modifications

Figure 1.6 on page 12 shows that the most frequent actions taken by students to improve or change models were the adding, deleting, and editing of training data. Students’ written work reveals that they typically performed these actions for rational and predictable reasons. Students added new training data to correct errors or to make a model’s behavior more complicated. They deleted subsets of examples to address particular problems in a model, for example deleting the last round of training examples recorded when a retrained model did not perform as anticipated, or deleting all training examples that affected a parameter whose model was not performing well. They cleared the training dataset when they gave up on fixing a model and wanted to start anew; one student wrote: “I found it often easier to just start over again when certain things were not working. When recording more examples, I often found that the training got even more muddled.” Students who did change the learning algorithm changed it when they thought it should be possible to build a more accurate model from data.

In addition to modifying the training data to attempt to build models that better met their goals for the system, students also often modified the data to reflect changes to their goals, as we discuss next.
1.4.5 Human Learning and Adaptation

Changing Goals

Students often adapted their goals for what they wanted the Wekinator to learn based on what they discovered through direct evaluation. One reason for this adaptation was that, through experimentation with the Wekinator, they realized that it was too difficult or impossible to build a model that conformed to their initial ideas about how they wanted gestures to control sound. When this occurred, students sometimes simplified the learning problem, for example by reducing the number of classes in Part B or changing the gestures themselves to be more easily differentiable. One student actually changed from using the webcam color tracker to using the trackpad input, which she understood better, so that she could better control the learning problem she presented to the Wekinator.

Another reason that students changed their goals for the models was that, through hands-on experimentation with the system, they sometimes discovered models performing in unexpected ways that they actually liked better than how they had planned for the system to work. One student described this experience in Part B: “...[A]s I was trying to train [the model], I began to get a model that would play beats depending on the extremity of the tilt. Beat 0 would be stationary, beat 1 would then be a slight tilt in any direction, and beat 2 would be an extreme tilt in any direction. I hadn’t really thought to do this initially, but once the training to some extent got there on its own I decided to go with it.” Another student wrote in Part A: “First I wanted to make something that worked just how I wanted, but then I thought it more important to create something that was controllable but also generated awesome sounds I would never have thought of. In my estimation, this was a mission accomplished!”
Two students indicated that they chose control gestures to use solely through exploration, rather than starting from a set of gestures they were interested in the Wekinator recognizing. One student wrote, “I didn’t have a very clear idea in my head of how I wanted the sound to change as the tether moved, so I used a trial and error method. Eventually, I realized that there were certain gestures that I wanted to correspond to certain sounds, so I recorded more examples to train the system accordingly. As I noticed patterns starting to emerge in my tests, I recorded more examples to make these patterns reliable while using the tether.”

Through direct evaluation and retraining, students also learned about the tradeoffs associated with different variations of their models, and they used their judgement to pick a model with the set of tradeoffs that they liked the best. One student wrote, “I had originally tried to use all four beats. I first tried using a neutral position where the computer was not tilted at all, and later used a position where the computer was tilted to the left. I eventually decided to eliminate this fourth position, as every single time it did this, I would lose two or three of my other beats, and get a much less accurate model (usually a cross validation accuracy of 50%). I eventually forfeit the extra beat for a much more accurate model.”

Learning Effective Strategies for Interactive Machine Learning

Students also adapted their strategies for effectively performing machine learning, based on their observations of how their actions affected the trained models. When asked about their strategies for model building, ten students indicated that they had learned during their interaction with the software to provide training data that more clearly expressed their intentions. One student wrote, “In collecting data, it is crucial, especially in Motion Sensor, that the positions recorded are exaggerated (i.e. tilt all the way, as opposed to only halfway).” Another wrote, “I tried to use very clear examples of contrast in colour for the tracker. If the examples I recorded had
values that were not as satisfactory, I deleted them and rerecorded... until the model understood the difference.” Many students reported that they looked for outliers in their training examples and deleted them. Some students even learned to balance class proportions in the training set (“Each extreme of a parameter should be trained with roughly the same number of examples”). This is remarkable, given that none of these machine learning concepts were introduced or discussed in class.

**Learning Effective Gestures**

Finally, students adapted the ways in which they interacted with the trained models, based on which gestures yielded the best results. Students’ comments indicated that they did not necessarily view models’ “musical expressivity” and “reliable classifications” as intrinsic properties of the models themselves; rather, these qualities had to do with the extent to which the students could learn to make musically expressive and reliably classifiable gestures using the models. Direct evaluation allowed students to find gestures that created the desired sonic outcomes, “practice” these gestures, and evaluate the extent to which a performer could learn to play the mappings well. Many of the students in the class were proficient musicians (outside of PLOrk), and their written comments revealed that they treated the process of learning to effectively play a model as similar to the process of practicing a traditional musical instrument, and that they viewed human effort and adaptation to be appropriate and necessary in both cases. For example, comments included: “The more time I spend with the instrument, the more I know how to control it” and “Practice is useful when learning an instrument.”

**1.4.6 Success in Using the Wekinator**

As shown in Figure 1.4 on page 11 students agreed that they were able to successfully create musically expressive instruments and reliable gesture classifiers using the
Wekinator. In general, the feedback from students in the written component of the assignment indicated that they enjoyed using the Wekinator and found it useful. Their comments included: “Learning by experimentation was a lot of fun!”, “I very much enjoyed this project!”, “I love Wekinator!”, and “It’s so cool, the Wekinator rocks.” Furthermore, students were able to build models that they were happy with in just 27.1 minutes in A and 16.1 minutes in B, on average. This is an extremely short amount of time compared to what would be required to build a reliable classifier or expressive continuous mapping by writing code in ChucK or Max/MSP, for example, even for an expert programmer.

A few aspects of the Wekinator were confusing or frustrating to some students. Some students did not realize that they could limit the features selected for each model or restrict the parameters affected by a training example, so they relied on encoding their goals for independencies among parameters and between features and parameters through carefully designing the training set (a difficult feat). Several students also expressed frustration that the learning algorithm they used did not create accurate models from the data. In particular, students using AdaBoost.M1 boosting on decision stumps encountered many difficulties creating accurate models for multi-class problems; these difficulties are predictable from the perspective of someone familiar with the algorithms, but students did not necessarily know that changing the algorithm would likely fix many of their problems. Apart from these issues, which can be addressed through interface improvements and better educating users about how to use the Wekinator, students did not present complaints or criticisms about the software.

1.4.7 Effects of Task and Order

To examine effects of task (A or B) and order (A first or B first) on the model building process, we conducted a two-way ANOVA with one within-subjects factor
(task) and one between-subjects factor (order)². We examined the total number of model retrainings, the total time spent on the task, and students’ agreement that they could reliably predict the sound the model would make for a given input gesture and that the Wekinator learned what they wanted it to (i.e., ratings for statements S3 and S4 in Figure 1.4). There was no significant effect of task on number of retrainings ($F_{1,19} = 1.08, p > .05$), total time ($F_{1,19} = 2.46, p > .05$), or S4 rating ($F_{1,19} = 0.21, ns$); there was a significant effect of task on S3 rating ($F_{1,19} = 12.04, p < .005$). There was no significant effect of order on number of retrainings ($F_{1,19} = 0.29, ns$), total time ($F_{1,19} = 1.46, p > .05$), S3 rating ($F_{1,19} = 1.19, p > .05$), or S4 rating ($F_{1,19} = 0.93, ns$). We discuss these findings in Section 1.5.2.

1.5 Discussion

1.5.1 The Wekinator as a Teaching Tool

As discussed above, students were generally very successful in achieving the assignment goals of building musical and reliable models, and they did so relatively quickly. In this regard, the Wekinator was a highly successful pedagogical tool: no other software system could have been used to allow students to build these types of musical gestural classification and regression systems, without requiring them to have significantly more expertise in machine learning and requiring them to write a great deal of code. By enabling students to design their own classifier- and neural network-based systems very quickly, the Wekinator allowed students to experience new ways interacting with computer music systems, as well as to explore the musical consequences of their design decisions.

²Because the order was unbalanced with 21 students, we used SPSS to compute a Type III SS ANOVA.
The Wekinator was a useful teaching tool in other assignments throughout the semester. All students used the knowledge gained in the assignment discussed in this chapter to create successful midterm performance pieces using the Wekinator. In the midterm performance, students worked in groups of two or three and built new interactive, gesturally-controlled instruments, then composed pieces for these instruments and performed them for their classmates and other Princeton students in a concert. Some of the more creative new instruments built using the Wekinator for the performance included a tether-controlled tubular bell synthesizer, a joystick-controlled mandolin chord generator in a laptop re-adaptation of Gershwin’s “Summertime” (Figure 1.9), and an accelerometer-controlled algorithmic glockenspiel process, where the Wekinator affected the glockenspiel tempo and volume. Two groups of students also chose to use the Wekinator in the final course project, which was left open-ended with regard to the nature of the performance or installation that students created and the hardware and software tools they used. In one of these projects, pictured in Figure 1.10, the students created a “musical petting zoo” of instruments including a joystick harmony-generator and a “tether harp” that played notes when the strings were plucked. The students invited bystanders to play with these instruments, then taught groups of people to collaboratively perform a simple piece.

In addition to providing a tool for students to effectively build new instruments and installations, several students commented in their written work later in the semester that they found the Wekinator useful as a creative tool for discovering and realizing new uses of the computer in their projects and compositions. For example, excerpts from the midterm performance reports include: “Generated mappings can produce interesting and unique sounds that you never would have thought of on your own, which can later be used to great effect,” and “Sometimes, too, interesting things that you didn’t intend happen ‘between the cracks’ of the examples you trained the model with. This can lead to some neat musical effects.”
Figure 1.9: Three PLOrk students performing their adaptation of George Gershwin’s “Summertime” from *Porgy and Bess* during the midterm concert. The leftmost performer is playing the melody using laptop tilt, the middle player is using the Wekinator to control mandolin chords with the joystick, and the rightmost player is using the Gametrak tether controller with the Wekinator to play an FM-synthesis drone.

Figure 1.10: The “Musical Petting Zoo” final course project in action. These visitors to the project exhibition are learning to play a Wekinator-based joystick harmony generator and a Gametrak tether harp.

The Wekinator was used throughout the semester as a tool for teaching modular design, object-oriented programming, and signal processing. In the midterm assignment, several groups of students chose to implement their own synthesis classes. By implementing the Wekinator ChucK SynthClass interface (see Section ??), they were
able to use the Wekinator to control new synthesis algorithms and compositional processes that they designed themselves. Also, in an assignment later in the semester, all students used their signal processing knowledge gained in the course to build a ChucK audio feature extractor that implemented the CustomFeatureExtractor interface (see Section ??).

Involving the students in designing their own code components to plug into the Wekinator not only allowed them to discover through practice why modular design and APIs are useful; it also gave them a quick way to experiment with their own feature extractors and synthesis algorithms, discover whether or not they worked correctly, and create something musically interesting with them without writing any additional code. Because the Wekinator was used so extensively in the course, some students expressed that they felt limited by the small number of available, pre-written synthesis algorithms available for them to use. Therefore, in future classroom use of the Wekinator, we plan to engage students themselves in building more synthesis classes and making them available to other students and all Wekinator users in an online repository.

As machine learning was not itself a primary focus of the course, we did not directly evaluate how much students learned about machine learning during the assignment or during their use of the Wekinator. However, as we discussed above, some students did learn a surprising number of fundamental and subtle machine learning concepts, including the circumstances under which some algorithms worked better than others, and how properties of the training data such as noise, inter-class and intra-class variability, and class imbalances affected the trained models. They learned this information not from discussion in class, but simply by interacting with the system and observing how different interactions and datasets produced different models. Therefore, using interactive machine learning seems like a potentially useful and fun
way to explicitly teach students about machine learning principles and algorithms, and this is another potential future application of the Wekinator.

1.5.2 Interactive Supervised Learning and Task Type

As discussed in Section 1.4.7 on page 24, students’ level of agreement with Statement S3 was the only measured quantity where a significant difference was found between Part A and Part B: students more strongly agreed in Part B than in Part A that they could reliably predict the sound the model would make for a given gesture. According to the students’ written responses, some of them intentionally created models with some built-in unpredictability in Part A, because they felt such models were more interesting or musical. In Part B, on the other hand, unpredictability seemed to be in competition with the primary goal of the task, creating a reliable gesture classifier. Future work might more deeply investigate the differences in interactive and algorithmic needs of users with different priorities regarding predictability versus reliability.

Based on the students’ written discussion of their strategy and evaluation methods, we believe there were other important differences between the way students built and evaluated the systems, and perhaps differences in how to improve the Wekinator to better support different types of tasks. The extent to which students had clear and fixed ideas about the gestures and gesture-sound relationships they wanted to learn did seem to be important in determining the strategy that they used to build the models. Several students with clear goals for the models went to great lengths to carefully design the training dataset to represent the precise gesture-sound relationship that they wanted, and it was students in Part B with clear ideas about the mapping they wanted who changed the learning algorithms or expressed frustration that the classifier they were using did not work. While more students in Part B than Part A expressed having clear ideas about the mapping they wanted before start-
ing to use the Wekinator (19 in B, 8 in A), many students in B ended up changing those ideas once they started working with the system; so in this study, the extent to which students had fixed ideas they wanted the Wekinator to learn did not break down cleanly by task. Students’ written responses regarding the extent to which their strategy remained fixed were not clear enough to allow testing for significant differences in their behaviors. Additionally, it is unclear why different students had such different goals for the system: this could be a matter of demographics (e.g., their academic major), aesthetic priorities, or other factors.

We believe that future work might explicitly compare differences in how users execute the interactive building of models when they have fixed versus more flexible goals, and future work on improving interfaces for interactive machine learning might consider the needs of these types of users separately. For example, an interface that can provide feedback to a user indicating that a different learning algorithm might be more appropriate for the current training dataset might be more valuable to a user with fixed goals who has more limited ability to change the training set. We discuss these ideas further in Chapter ??, where we present work with a user whose goals for the trained models are relatively fixed by musical conventions, and in Chapter ??.

1.5.3 Improving the Wekinator

Students’ written comments highlighted several aspects of the Wekinator that might be improved to make it more usable by people who are machine learning novices. First, as mentioned above, several students indicated frustration that their models were incapable of learning more than two or three classes. Examining the logs of their work, it became clear that these students were using AdaBoost.M1 boosting on decision stumps. This algorithm will perform poorly in certain simple classification problems, such as recognizing a laptop’s tilt forward, backward, left, and right. It seems that the students used AdaBoost.M1 because it was the first learning algorithm in
the alphabetically-organized list in the Wekinator GUI, and they used decision stump base learners because that was the default setting for the AdaBoost.M1 algorithm. In response, we modified the Wekinator so that the default setting for AdaBoost.M1 is now to boost on decision trees, which do not suffer from the limitations of decision stumps on multi-class problems. The question still remains, though, of how to most effectively educate novice users about which algorithms might perform best for their data, or to inform them about how they might change the parameters of an algorithm to achieve better results. At the very least, offering the user the suggestion that he might experiment with changing the learning algorithm if he is having problems might circumvent some of the frustration experienced by the students.

The question of how to educate novice users about certain machine learning concepts that might be helpful to them is also relevant to other aspects of the system, including interfaces for feature selection, adding meta-features, and computing training and cross-validation accuracy. It was clear from students’ comments that some of them did not understand how to use or interpret one or more of these components of the Wekinator. However, we believe that machine learning novice users could be better educated about these concepts through careful explanations within the user interface and/or a “help” system or tutorial, and that these users should be able to use the Wekinator effectively for many problems without having to acquire too much theoretical knowledge about machine learning. In the near future, we hope to collaborate further with target novice users, such as students and professional composers, to create more helpful documentation.

1.5.4 Further Discussion

We further discuss the implications of this work regarding algorithm and interface design, the use of interactive machine learning by novices and in creative work, and the larger role of interaction in supervised learning in Chapters ?? and ??.
1.6 Conclusions and Future Work

In this work, we have found that the Wekinator enables students to successfully and quickly build musically expressive and accurate models, and observed that students relied greatly on being able to evaluate models in a hands-on way and modify models’ training data in order to assess models’ subjective quality and improve them over time. We have also gained a valuable perspective on how the Wekinator can teach users to develop effective strategies for interactive machine learning, and how interaction can over time help users develop and change their goals for the interactive systems they are creating, as well as allow them to practice interacting effectively with the models they have built.

This work suggests several veins of further research, including studying how the flexibility of users’ goals for supervised learning might impact their interface and algorithm requirements, how to better support interactive machine learning by users who are machine learning novices, and how to integrate hands-on interaction into the teaching of machine learning concepts. This work also suggested several improvements to the Wekinator, some of which have been implemented and some of which are planned for the near future.

Although the Wekinator was not developed as a teaching tool, our experiences using it in the classroom have been remarkably positive: students were able to use the Wekinator to create projects that would not have been possible otherwise, and they were able to learn about a variety of topics, including machine learning, object-oriented programming, sound synthesis, and gestural analysis in a hands-on manner that engaged their creativity. At the same time, observing students using the Wekinator and hearing their feedback about the software was highly informative to us, not only suggesting how the software could be improved, but also suggesting new research questions. In our future work with the Wekinator, we hope to continue to create and take advantage of these synergies between teaching and research.
Bibliography


