Human Model Evaluation in Interactive Supervised Learning

Rebecca Fiebrink[1,2], **Perry R. Cook**[1,2], **and Daniel Trueman**[2]
Departments of [1]Computer Science and [2]Music
Princeton University
Princeton, New Jersey, USA
{fiebrink, prc, dtrueman}@princeton.edu

ABSTRACT

Model evaluation plays a special role in interactive machine learning (IML) systems in which users rely on their assessment of a model's performance in order to determine how to improve it. A better understanding of what model criteria are important to users can therefore inform the design of user interfaces for model evaluation as well as the choice and design of learning algorithms. We present work studying the evaluation practices of end users interactively building supervised learning systems for real-world gesture analysis problems. We examine users' model evaluation criteria, which span conventionally relevant criteria such as accuracy and cost, as well as novel criteria such as unexpectedness. We observed that users employed evaluation techniques—including cross-validation and direct, real-time evaluation—not only to make relevant judgments of algorithms' performance and interactively improve the trained models, but also to learn to provide more effective training data. Furthermore, we observed that evaluation taught users about what types of models were easy or possible to build, and users sometimes used this information to modify the learning problem definition or their plans for using the trained models in practice. We discuss the implications of these findings with regard to the role of generalization accuracy in IML, the design of new algorithms and interfaces, and the scope of potential benefits of incorporating human interaction in the design of supervised learning systems.

Author Keywords

Interactive machine learning, evaluation, gesture, music

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction Styles*; I.2.6 Artificial Intelligence: Learning

General Terms

Design, Human Factors

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

INTRODUCTION

Machine learning offers a set of powerful algorithmic tools for understanding, modeling, and making decisions from data. Application of these tools has been critical to advances in domains as diverse as bioinformatics, information retrieval, gaming, robotics, and beyond. Along with a growing number of human-computer interaction researchers, we are interested in the question of how to leverage human interaction and expertise in new ways to make machine learning even more effective and useful in more application domains. We are particularly interested in engaging human interaction throughout the process of building a working machine learning model, by enabling the user to iteratively evaluate the current state of the model and take appropriate actions to improve it.

Human model evaluation plays a critical role in an interactive machine learning context: evaluation methods are useful not only for informing the user about the relative or absolute quality of a given model, but also for informing the user how he or she might take action to improve it. As we seek to build effective and usable interactive machine learning systems, it is crucial to understand users' own evaluation criteria for models, the ways users engage different evaluation techniques to gain the information they need, and the ways users employ this knowledge to shape their future interactions with the system. This understanding can be used to inform the design of user interfaces for model evaluation, the choice and interpretation of model evaluation metrics, and the selection and design of learning algorithms themselves. It can also lead to a greater appreciation for how applied machine learning applications may stand to benefit from incorporating human interaction. It is precisely this understanding that we have sought in this work, through three studies observing end users employing interactive machine learning to build real-world gesture analysis systems in music.

Our most significant findings are summarized as follows: Users' criteria for model evaluation included more than just correctness, encompassing subjective judgments of properties such as cost, decision boundary shape, confidence, and complexity (which, notably, was sometimes highly desirable). The relative importance of these criteria and the way they were defined varied appreciably across applications. A hands-on, fine-grained, exploratory evaluation of models allowed users to judge them against these criteria and to learn

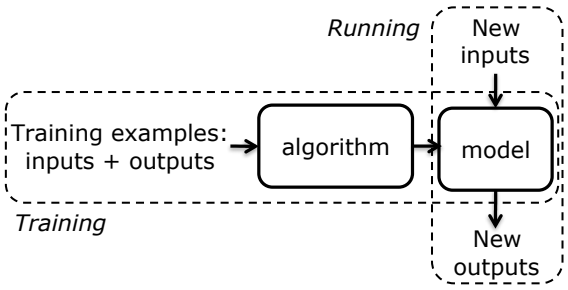

Figure 1. A supervised learning algorithm creates a model from training data. The trained model can then compute new output values from new inputs.

the information they most needed to improve models' behaviors. Cross-validation was sometimes useful for quick comparison and validation, but even when generalization accuracy was of high importance to the user, cross-validation accuracy did not necessarily correlate positively with the user's subjective rating of a model's quality. Finally, model evaluation also provided feedback to users that helped them learn to interact more effectively with the supervised learning process and to interactively discover and manage tradeoffs between building models that were easy to create and building models that were most useful in practice.

BACKGROUND AND MOTIVATION

Supervised Learning

This work concerns a family of machine learning and pattern recognition algorithms known as supervised learning. Here we will give only a very basic overview of supervised learning; a more thorough treatment can be found in [6], which is the reference for our overview unless otherwise indicated. A supervised learning algorithm is essentially a tool for producing a mathematical model or function that, given some input, produces some output. The algorithm infers, or *learns*, this model from a training dataset, which is a collection of data points or "instances" consisting of example inputs paired with their corresponding outputs, or "labels." After this model is built, it can compute new output values for new inputs, even for inputs not present in the training set. Figure 1 illustrates the relationship between the data, algorithm, training, and model.

The inputs to a model are vectors of numbers, or features. For example, a model of human gesture might take in a feature vector extracted from sensors worn on the body, where the features themselves are raw sensor values, statistics computed on those sensor values, or both. The output of a model is either a real number or one of a finite set of discrete labels, depending on whether the model is for regression or classification. For example, classification could be used to build a sign language alphabet classifier that outputs the written word a human is signing, as in [1]. On the other hand, regression could be used to build a continuous gesture-to-speech controller, such as one component of Fels and Hinton's Glove-TalkII, in which continuous hand motions result in gradually changing articulatory control over a speech synthesis algorithm [8].

A primary goal of supervised learning is to model the relationship between inputs and outputs in the training set in a way that allows *generalization*, producing reasonable outputs for new inputs not present in the training data. Concern for generalization is at the core of the theoretical analysis and design of many machine learning algorithms. In the PAC-learning framework, for example, a learning algorithm is by definition capable of classifying instances not (necessarily) in the training set with a high accuracy rate, with a high probability [21]. Widely-used algorithms such as AdaBoost and support vector machines are designed to explicitly to maximize generalization accuracy (that is, the proportion of future inputs that are assigned the correct label) [6].

An emphasis on generalization accuracy underlies many standard metrics used to evaluate the suitability of an algorithm (or parameterization thereof) for modeling a given dataset. Unless one knows the identity of all future inputs to a model and their proper labels (in which case building a perfect model is trivial), generalization accuracy must be estimated from the available data. Using accuracy on the model's training dataset is a poor estimate, as this can assign too favorable a rating to a model that has "overfit" to the training data and is poor at generalizing. So, the available data can be partitioned into a training set and a mutually exclusive test set for evaluating performance. Cross-validation is a commonly-used technique that repeats this procedure several times such that each available instance is present in the test set of a single iteration, then averages test set accuracy over all iterations. Alternative measures such as F-measure, cost, precision, recall, and area under ROC may be used under certain circumstances, but in each case, the goal remains to estimate the model's relevant future behavior from the available, finite data. In order to do this, it is standard for the learning algorithms and evaluation procedures to make the basic assumption that the future inputs to a model will look more or less like the training data. This can be stated probabilistically as the assumption that the training data and the future data examples are both sampled i.i.d.[1] from a shared distribution over the data space.

Interactive Machine Learning

Our work focuses on supporting human-computer interaction in the context of creating machine learning systems, where users are engaged in tasks including choosing and training a learning algorithm, evaluating and comparing models, and supplying training data. The scope of relevant users includes researchers applying machine learning techniques to data analysis in an application domain of their expertise, developers of user interfaces that contain machine learning components, and end users of software tools that directly engage the user in controlling some aspects of a machine learning system, such as providing training data and evaluating trained models.

There is an exciting thread of recent research investigating how human interaction can be leveraged in new ways in the

[1] independent and identically distributed; i.e., having a shared underlying probability distribution and being mutually independent

creation of machine learning systems. For example, ManiMatrix allows users to manipulate a confusion matrix to interactively steer a model's performance to reflect their priorities [16], and EnsembleMatrix harnesses human users to optimize ensembles of learners [20].

One significant opportunity for improving supervised learning systems using human interaction lies in enabling the user to evaluate a model, then edit its training dataset based on his or her expert judgments of how the model should improve. This approach was proposed under the term "interactive machine learning" by Fails and Olsen [7], whose Crayons system allows users to improve an image classification model by iteratively evaluating the system, providing new training data, and retraining. We have found this approach very useful in building systems for audio analysis [12] and musical gesture analysis [9]. This general approach has also been applied to handwriting analysis in CueTip [19], web image classification in CueFlik [13, 2, 3] (where it is termed "end-user interactive concept learning"), document analysis by Baker et al., [4], and sensor-based interaction design in Exemplar [15]. This body of work suggests that, for domains where it is feasible to provide additional training data, and where a human user has knowledge of the modeling problem, user modification of the training data allows a natural and effective means to take advantage of user expertise.

Motivation

We propose, along with prior work such as that of Amershi et al. [3], that interactive machine learning interfaces must not only enable the user to effectively edit the training data, but also supply him with the information he needs to most effectively guide the machine learning process. One aim of our work is to learn more about what this information entails, based on users' own criteria for evaluating models and the actions they take based on their evaluations. Additionally, in any machine learning application, the choice of algorithm and objective evaluation metrics reflect certain assumptions about the goals for the trained model, for example the goal of good generalization accuracy. A second aim of our work is therefore to interrogate how knowledge of end users' evaluation criteria and evaluation strategies can inform the selection or design of learning algorithms and evaluation metrics. Finally, though interactive machine learning of this type has not been widely studied in gesture analysis or in real-world applications, several of the users we have worked with have found it to be a transformative technique in their work in music. So, a third goal is to provide an enriched understanding of the merits of an interactive approach to supervised learning by studying users applying it to real-world problems in this domain.

APPLICATION DOMAIN, SOFTWARE, AND STUDIES

Application Domain and Software

We have observed and worked with users applying supervised learning to different human gesture modeling problems in computer music composition and performance. Some gesture analysis applications in music entail the identification and classification of discrete control gestures, in order to initiate computer audio or visual events (e.g., [18]). Certain of these applications closely resemble gesture analysis problems in the design of interfaces for general-purpose control (e.g., [23]) and gaming (e.g., [14]). Other applications in music involve the analysis of acoustic performers' musical gestures, for example analysis of the bowing articulations of a string player (e.g., [24]). These applications have certain parallels to analysis of natural human motion, for example in sports or medical diagnosis (e.g., [5]). A third category of applications that has no clear counterpart in non-creative domains is the design of new computer music instruments, where the trained model provides a function mapping from the sensed gesture space into the parameter space of an audio synthesis algorithm (e.g., [17]). There, the goal is to allow a human performer to "play" the synthesis algorithm expressively in real-time, not to model human gesture "correctly," per se. Each of these application categories is represented in at least one of our three studies.

Gesture modeling is a natural domain for applying interactive machine learning: when the goal is to build a gesture model of the user herself, it is easy to obtain representative training and evaluation data and to trust that she understands the goal of the modeling problem well enough to make reasonable assessments of model quality. However, despite widespread application of supervised learning to gesture modeling in music, previous work does not incorporate human interaction in this way.

The software used in all three studies, the Wekinator, has been designed to facilitate the interactive application of standard supervised learning algorithms in real-time domains, including music. The software is described in detail in [9]. It supports an iterative approach to IML illustrated in Figure 2, and it provides user interfaces for the creation, editing, and visualization of the training data, the selection of machine learning algorithms and their parameters, and the running of trained models on real-time inputs. Users are able to create and add to the training dataset by demonstrating different inputs (here, physical gestures) in real-time while providing the corresponding classification or regression labels using the Wekinator GUI. Users can also run trained models in real-time, producing a stream of real-time model outputs in response to an incoming stream of gestural inputs. In this regard, the Wekinator is similar to the Exemplar system [15], though the Wekinator is a more general-purpose tool that includes support for experimenting with different algorithms and applying machine learning to various real-time problems, including the analysis of audio and other non-gestural inputs.

The Wekinator includes standard algorithms for classification (AdaBoost.M1, J48 decision trees, k-nearest neighbor, and support vector machines) and multilayer perceptron neural networks for regression. While the Wekinator can take inputs from any external feature extractor, it includes several simple, built-in gestural feature extractors that allow users to provide gestural inputs using the webcam, the laptop's internal accelerometers, the trackpad (used as a 2-dimensional finger position sensor), and USB human interface devices (HIDs) including standard game controllers.

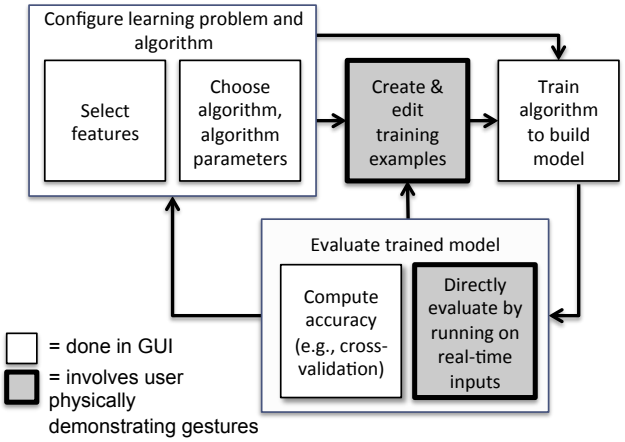

Figure 2. The interactive machine learning workflow supported by the Wekinator, where white actions are performed using the GUI and grey actions are performed through real-time demonstration.

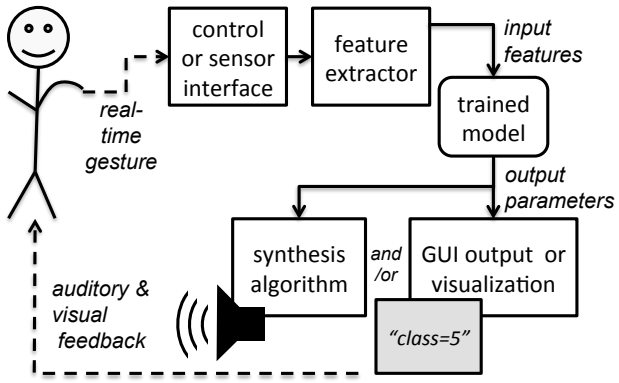

Figure 3. Direct model evaluation in the Wekinator. The user supplies new gestural inputs in real-time and evaluates the model's response by observing the sonic or visual processes controlled by the model's outputs.

In the Wekinator, users can evaluate trained models both by computing cross-validation and by hands-on, or "direct" evaluation. In direct evaluation, illustrated in Figure 3, the user provides inputs (here, gestures) in real-time and observes the model's behavior. The user can view the model's current outputs as text fields in the GUI, or the models' outputs can be sent over the Open Sound Control protocol[2] to another program, for example to control a visualization or sound synthesis algorithm in real-time.

Studies

We have conducted three studies of people applying supervised learning to their work in computer music. In the first study ("A"), we led a user-centered design process (see [22]) with seven composers, which focused on the refinement of the Wekinator. The participants were six PhD students and one faculty member in Princeton's Music Composition department. Participants met weekly for three hours each week for ten weeks. During each session, they discussed how they were using the software in their work, proposed improvements, asked questions, and experimented with the software. We took written notes of composers' questions, suggestions, and discussion topics. In between meetings, we implemented suggested improvements. After the ten meetings, participants completed a written questionnaire about their experiences in the process and their evaluation of the software.

The composers in Study A primarily used the Wekinator to build new musical instruments, creating neural network models that input the sensed human gestures and output control parameters that drove a digital audio synthesis algorithm in real-time. Participants used a variety of gestural input devices, including webcams, laptop motion sensors, HID devices, and custom-built sensor arrays. The two most frequently used synthesis algorithms required the setting of several (nine or more) real-valued control parameters, each of which had highly nonlinear and interdependent effects on the sound. Composers had found these algorithms difficult to control in a musically satisfying way using either a GUI or an explicitly programmed control sequence. More information about this study and a discussion of how participants used supervised learning in composition is published in [11].

In the second study ("B"), we observed a group of 21 students using the Wekinator in an assignment focused on supervised learning in interactive music performance systems. All students were enrolled in an interdisciplinary computer science and music course that included a significant emphasis on interactive performance technologies. Students ranged from first through fourth year in undergraduate study, they came from a variety of majors, and most had only rudimentary knowledge of machine learning. Prior to the assignment, the students received an in-class discussion and demo of the Wekinator software. In the assignment, each student was asked to use an input device (USB controller, motion sensor, trackpad, or webcam) to create two gesturally-controlled music performance systems, one that employed a classifier to trigger different sounds based on each gesture's label, and one that employed a neural network to create a continuously-controlled musical instrument (similar to Study A). These tasks were assigned in preparation for the midterm concert, which required students to compose and perform pieces using interactive systems that they had built (though not necessarily the systems built during the assignment). The software logged the students' actions during both tasks. The assignment also included 12 short-answer questions about the process of creating the two systems. Students were graded on completing the assignment and answering the questions thoughtfully.

The third study ("C") was a case study in which we worked with a professional cellist/composer to build a gesture recognition system for a sensor-equipped cello bow. This bow, called the "K-Bow,"[3] contains sensors for measuring the position and motion of the bow in real-time, including acceleration along three axes, tilt, horizontal and vertical position of the bow relative to the instrument, hair tension, and grip pressure. The goal of our work with the cellist was to build a

[2]http://opensoundcontrol.org/

[3]http://www.keithmcmillen.com/products/k-bow/

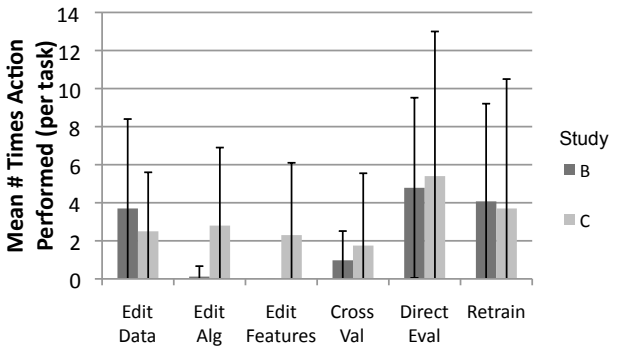

Figure 4. The mean number of times each interaction was performed, for each task and user in B, and for each task in C.

set of gesture classifiers to process the bow sensor values in real-time and produce musically appropriate labels. These labels could then be sent to the cellists' own composition programs in Max/MSP (a standard music composition environment) and used to influence computer-generated sound and/or visuals. Following a preliminary study with the cellist to develop infrastructure for communicating the bow outputs to the Wekinator and to choose gestures to classify [10], in Study C we worked with the cellist to construct eight robust gesture classifiers that labeled the bow's vertical position, horizontal position, roll, speed, articulation style, direction, position on or off the string, and grip squeeze strength. Throughout this process, the software logged all actions and saved all iterations of the trained gesture models. We also solicited the cellist's subjective rating of each model's overall quality, and we recorded observations of the model-building process using video and written notes.

FINDINGS

Interactive, Iterative Model-Building

Participants in all three studies took an approach to model-building in which they interactively created the training data from scratch, then iteratively built models, evaluated the models, and modified the models by changing the training dataset, learning algorithms, selected features, and/or algorithm parameters. Students in Study B retrained the algorithm an average of 4.1 times per task ($\sigma = 5.1$), and the cellist in C retrained an average of 3.7 times per task ($\sigma = 6.8$). In between retrainings, users most frequently changed the model behavior by editing the training data. Figure 4 summarizes the logged actions of users in B and C. For Study A, participants' questionnaires indicated that they also iteratively retrained the models, and they almost always chose to modify the models only by editing the training dataset. In all studies, retraining of the models was nearly always fast enough to enable uninterrupted interaction with the system (i.e., a few seconds or less).

The Use of Cross-validation

In Study A, composers never used cross-validation. In B and C, cross-validation was used occasionally; on average, students in B used it 1.0 times per task ($\sigma = 1.5$), and the cellist in C used it 1.8 times per task ($\sigma = 3.8$).

The five students in Study B who commented on their use of cross-validation indicated that they treated a high cross-validation accuracy as reliable evidence that a model was performing well. 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 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. [3], who observed that users felt pressure to optimize cross-validation accuracy as an end in itself, rather than using it as an informative tool.

In Study C, cross-validation was used to quickly and objectively compare alternative classifier algorithms on the same dataset. This was done either after direct evaluation had shown that a reasonably good model had been built from the current training dataset, and it was uncertain which algorithm might perform most accurately, or when direct evaluation had revealed the learning problem to be particularly stubborn, and many different algorithms and feature selections were tried in succession to see if any one 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; $\sigma = 1.5$).

The Use of Direct Evaluation

Participants were also able to perform direct, hands-on evaluation to assess model performance. In direct evaluation in Studies A and B, models' real-time outputs drove sound synthesis algorithms. In C, the cellist had the option of displaying outputs textually in a GUI or controlling a live visualization, as discussed below.

Participants in all three studies used direct evaluation more frequently than cross-validation. Participants in A only used direct evaluation; participants in B performed direct evaluation an average of 4.8 ($\sigma = 4.8$) times per task, and the cellist in C performed direct evaluation 5.4 ($\sigma = 7.6$) times per task. (Note that C required a minimum of one direct evaluation per retraining, as the cellist was required to assign each model iteration a subjective rating based on her evaluation.) Compared to cross-validation, direct evaluation was used under a wider variety of circumstances and to evaluate the model against a wider array of subjective criteria. In the following section, we discuss the criteria users employed when directly evaluating models, and how users took action to improve models against these criteria.

Direct Evaluation Criteria

Correctness

Predictably, direct evaluation was employed to systematically check a model's correctness over a range of the input space of gestures. In all studies—including the expressive instrument design projects in A and B, where there was no objectively right or wrong model behavior—users identified the model's behavior as incorrect when it produced an output contrary to what they believed was appropriate and ex-

pected. Users typically reacted to this finding by modifying the training dataset—adding new inputs similar to those that caused the error, but with the correct labels—and retraining. In extreme cases of failure, users deleted the training set and recreated it from scratch.

Cost

Our observations suggest that users held an implicit error cost function that variably penalized model mistakes based on both the type of misclassifications (i.e., the user's label and the model's label) and their locations in the gesture space. For example, the cellist in C verbally indicated that classification mistakes a human cellist might easily make were less problematic. Another concern for users in A and B was whether a model could produce desirable or correct outputs for the types of gestures that would be used in a performance; the model's behavior on gestures not used in performance was inconsequential. Note that, for these users, the goal of applying supervised learning was not to model a gestural vocabulary that was strictly defined a priori: the users had considerable leeway in choosing which gestures would be used in performance. One student in B wrote about his strategy for designing the training set: "I tried to assess which [sounds] I would use more often and correlate them with [features] that were easier to obtain on the [input device]..." One role of direct evaluation was to search for input gestures that produced good outputs and practice them to ensure that they could be performed reliably. Several users in A and B likened this to practicing a traditional musical instrument, and just as in learning to play the violin, for example, users accepted responsibility for adjusting and improving their own performance-time actions to ensure good behavior of the model. In other words, users were not only seeking to improve the model to minimize cost, but they were also interactively attempting to adjust the cost function itself to favor the model.

In Study A, misclassification cost was also higher for examples similar to those in the training set: some composers indicated that they designed their training examples with the explicit intention of using the model to play those chosen sounds when they made the corresponding chosen gestures, and they were not happy when the trained model did not allow this. Composers were typically much less concerned with the model behavior matching their pre-formed expectations for input gestures dissimilar to those in the training set.

Decision Boundary Shape

In Study C, 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 bow 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.

Label Confidence and Posterior Shape

In Study C, the cellist also took model confidence into account when evaluating its quality. The cellist was proficient in Max/MSP, and she worked with us during the study to design several simple Max/MSP visualizations to help her understand more about the model during direct evaluation. One frequently-used visualization displayed the model's estimated posterior probability distribution over the set of class labels. When the model classified her current bow gesture correctly but also assigned relatively high posterior probabilities to several incorrect labels, the user expressed dissatisfaction (at one point exclaiming, "Come on, be more sure than that!") and attempted to improve the model's confidence, usually by adding more training data. 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.

Complexity and Unexpectedness

In Study A, composers valued models that produced sonically interesting parameterizations of the synthesis algorithm, which a human could manipulate over time in a musically sensitive way. Composers typically did not construct the models with a full set of physical and musical gestures already in mind; rather, a common strategy for building the models was to choose a few different and interesting synthesis parameter values that they wanted the instrument to be capable of playing, match each of these with a different gestural input in the training set, then directly evaluate the trained model to discover and explore the sounds that arose as they moved between and around the gestures present in the training set.

When evaluating the model using gestures outside the training set, two of the characteristics that were most important to composers were its complexity and unexpectedness. Unlike in a typical machine learning application, complexity was desirable, and composers sometimes added training data with the explicit intention of making the model more complex. Some composers remarked that the complexity of functions generated by the Wekinator's neural networks made their new instruments feel more like traditional, acoustic instruments, which by nature involve very complicated relationships between the physical gestures of a performer and the sound produced. Additionally, composers valued the fact that they could be surprised by the sounds generated by a gesture not in the training set, allowing them to find unimagined and compositionally useful sounds in the synthesis space.

Subjective Evaluation Versus Cross-validation

In Study C, every time an algorithm was trained or retrained, the cellist evaluated the model using direct evaluation and assigned it a subjective rating from 1 to 10. After the completion of the study, we computed cross-validation accuracy for all models from the logged data. 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 of each model and its cross-validation accuracy. Even though model accuracy was extremely important to the user, the data show that a model's cross-validation accuracy did not always positively correlate with its subjective rating. 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), while speed and articulation classification tasks had positive coefficients (0.65 and 0.93).

In the four tasks with negative correlations, the training sets of certain iterations provided representations of the learning problem that were both inaccurate and simple to model. 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, the 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, therefore, problems with the training set were undetectable using cross-validation. Direct evaluation allowed the cellist to discover the problems, fix them, and ultimately create models rated "10" for each task.

Evaluation and Human Learning

Cross-validation and direct evaluation served as feedback mechanisms to the users, enabling them to discover whether or not their recent changes to the training set or algorithms had had the desired effect on the retrained models. In fact, as the Wekinator did not provide any machine learning tutorials or hints, the feedback obtained from cross-validation or direct evaluation was the only mechanism for users to learn how their actions were likely to affect the system. In this way, evaluation actions trained the users, none of whom had significant experience or instruction in machine learning, to use the system more effectively.

Teaching Users to Provide Better Data

In Study B, 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 [input features]... If the examples I recorded had values that were not as satisfactory, I deleted them and rerecorded... until the model understood the difference." 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"), even though this more advanced machine learning concept was not introduced in class.

The cellist in Study C 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), 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.

Teaching Users What is Possible

Users often adapted their goals for the system based on what they discovered through direct evaluation. This was especially true for users in A and B, who had some choice in the types of models they asked the computer to learn. One reason for adaption was that, as machine learning novices, users did not have well-formed expectations of how different algorithms worked or what could be accomplished with supervised learning. When discovering that their efforts were failing to produce a model that worked how they wanted (for example, failing to create a neural network with a smoothly linear mapping between a gesture and a synthesis parameter), users adjusted their goals. Another reason for adaptation was that, through hands-on experimentation with the system, users discovered that models performed in unexpected ways that they actually liked better than their initial goals. Twelve students in B indicated that they changed their minds about which gestures they wanted to use after their initial efforts failed to produce a model that worked how they wanted, or when they discovered that a model did something unexpectedly useful for a new gesture. Two other students and the majority of the composers indicated that their strategy was to choose performance-time gestures to use solely through exploration, rather than start from a set of gestures they were intent on incorporating. In C, the cellist's goals were more subtly adjusted to reflect knowledge about what a model was able to learn alongside knowledge of what type of model would be most useful in practice. For example, after building a speed classifier that worked well for three classes, she decided to try building a finer-grained speed classifier for five classes.

Providing Feedback on Users' Gestural Techniques

In Study C, the cellist also gained a new perspective on her own bowing technique, when she discovered through consistently poor system behavior 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 riccocet 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 riccocet. After adjusting her technique, she was able to both train a model that performed better and produce a better cello sound.

Usability and Usefulness of the Software

Our findings indicate that participants understood how to use the software, found it useful, and were engaged in meaningful interactions during our observations. At the conclusion of Study A, composers highly agreed that the Wekinator allowed them to create more expressive models than other techniques (mean $= 4.5$ on a 5-point Likert scale) and to create models more easily (mean $= 4.6$). In Study B, students highly agreed that they were able to create models that learned what they wanted (mean $= 4.6$), that their classifiers provided reliable classifications (mean $= 4.9$), and that their neural network models were musically expressive (mean $= 4.1$). All students subsequently succeeded in employing the software in their midterm performances. In C, the cellist rated the quality of six of the eight final bow gesture classifiers as "10" and the other two as "9" on a 10-point scale. A more thorough evaluation of software usefulness and usability is provided in [9].

DISCUSSION

Model Evaluation in Interactive Contexts

Our observations suggest that metrics such as test set accuracy or cross-validation accuracy may be, on their own, insufficient for assessing the quality of models designed for interactive human use. First, users were concerned with evaluating more than the overall accuracy or error rate. Direct evaluation allowed them to identify where and how the model was likely to make mistakes, enabling them to make a cost-sensitive assessment based on criteria like error severity and the extent to which it might be possible to avoid using error-prone gestures during a performance. Direct evaluation also allowed users to evaluate models against highly subjective criteria, such as "unexpectedness" of model behavior.

Furthermore, in interactive machine learning, cross-validation or held-out test set accuracy may be problematic for other reasons. As demonstrated by the negative correlation between cross-validation accuracy and user rating for some tasks in Study C, the training set may be a particularly poor resource for estimating generalization performance during certain stages of the interactive model creation process, especially when the human has not yet discovered problems with the training data. Additionally, in this approach to IML, the user manipulates the training set as a means to directly influence the trained model's behavior, for example by adding properly-labeled examples to correct mistakes in error-prone areas of the input gesture space. Given that the user is consciously employing the training set to manipulate model performance, it is not necessarily reasonable to assume that the training examples will at all resemble the examples seen by the model in the future. Speaking probabilistically, it may be too strong an assumption to say that the training examples and the future examples are sampled i.i.d from a shared underlying distribution. So, evaluation based on a test set partition of the user-generated training data may not be meaningful for estimating model performance on future inputs. At the same time, in the case that the person driving the interactive machine learning process will also be the future user of the trained model, the user may know how to generate evaluation data that is more representative of future inputs.

Generalization Accuracy and the Choice and Design of Algorithms for IML

Supervised learning algorithms are often explicitly designed with the goal of maximizing generalization accuracy. Generalization accuracy was indeed a goal for many of our observed users, in that they intended the system to produce reasonably accurate outputs for gestural inputs that were not identical to those in the training set. In the music performance scenarios for which users were creating the models, there would be inevitable variation in future inputs due to human error and circumstances including room lighting and camera position when using the video input, or non-identical sensor calibrations when using the sensor bow. To some degree, the human user can mitigate the effects of changing circumstances by adding more training examples and retraining. But so long as the goal is to create a robust system for use in performance, the ability to accurately generalize to previously unseen inputs remains important.

As we have discussed, users were concerned with other model characteristics in addition to accuracy. When these other criteria are known and quantifiable, it may be useful to choose or design algorithms that explicitly consider these criteria during training. For example, algorithms might explicitly enforce decision boundary smoothness or function complexity. Alternatively, it may be useful to employ algorithms that expose parameters that users can employ to explicitly adjust models along dimensions that are important to them. For example, many algorithms can incorporate a regularization parameter controlling the degree to which overfitting is penalized [6], and exposing this parameter to the user could allow a means of interactively adjusting the complexity of the model.

Many learning algorithms are designed to produce a model with good generalization accuracy (as estimated, again, from the available training data), at the cost of good training accuracy. Even when generalization accuracy is of primary importance to the user, privileging generalization accuracy at the expense of training accuracy might be problematic in this type of interactive machine learning context. First, for the reasons discussed above, the training set may sometimes be a poor resource for estimating generalization accuracy. Second, the IML user relies on the algorithm's attention to the training examples as the primary means of influencing the model's behavior after training. By down-weighting the importance of training accuracy—that is, by being willing to misclassify portions of the training dataset—the algorithm may be ignoring an intentional attempt by the user to shape the model's behavior. It is possible, therefore, that training accuracy may play a more important role in interactive supervised learning than in conventional supervised learning. Future work might investigate whether algorithms such as

k-nearest neighbor, which does not explicitly address generalization accuracy and which can trivially achieve perfect training accuracy, are preferred by interactive machine learning users under certain circumstances.

The Benefits of Interactive Evaluation

Model evaluation provided a feedback loop that aided users in developing effective strategies for building working systems. Users were "trained" by the system to take appropriate actions to improve models according to their subjective goals, for example learning to provide clearer training examples. This feedback also helped users learn what was possible to accomplish with the given learning algorithms and features; in response, users sometimes adapted their interactions with a model during performance to work around its shortcomings. At other times, users made changes to flexible aspects of the learning problem, such as the number or nature of gesture classes to be used, in order to create models that were both feasible to build and most useful in practice. These behaviors bode especially well for the usefulness of interactive machine learning in other domains where users have some freedom in defining and changing the concept learned by the models, or where users are able to adapt their behavior to treat a trained model more "gently." In these cases, interactive design of the machine learning component can help users choose the most appropriate learning concepts and learn to provide good training data for those concepts.

Though previous work, notably that of Amershi et al. [2], has explored ways to provide the user with visual interfaces to efficiently evaluate trained models and choose new training examples, it is not clear how to extend this work to modeling problems for which there is no existing dataset of unlabeled examples, or in domains for which it is difficult to visualize examples. In this work, the exploratory nature of direct evaluation provided a means for users to assess the trained model's behaviors against a variety of subjective criteria, and the example-level granularity of the knowledge gained this way was critical to both making judgments of model quality and making corrective edits to the training data. Further work exploring how to guide users' evaluations to interesting or important areas of the model input space would be valuable. Additionally, it is worth exploring whether a gesturally-controlled, example-granularity exploration of the input space might be useful for directly evaluating models in domains where the model input itself is not human gesture.

CONCLUSIONS

We have observed people applying interactive supervised learning to several gestural analysis problems in computer music. User interactions included creating training data by real-time demonstration, evaluating models by computing cross-validation accuracy and by running them on new inputs in real-time, and iteratively improving models by changing aspects of the training data, algorithm, or features, and retraining. These interactions are appropriate for domains in which the user is capable of efficiently supplying training data and competently judging the suitability of the trained models for the application context.

In this work, we examined the criteria users held for assessing the quality of trained models, the techniques they employed to evaluate models, and the ways that model evaluation informed their goals and interactions with the system. We observed that cross-validation was sometimes useful for quickly assessing accuracy, but direct evaluation allowed users to make accuracy assessments that were cost-sensitive, assess models against a wider set of criteria, and detect problems with inaccurate models that were undiscoverable using cross-validation alone. Users employed cross-validation and direct evaluation to inform their immediate next actions with the system—for example, the choice of a certain algorithm, or the addition of particular training data. Model evaluation also provided feedback that enabled users to develop more effective interaction strategies for guiding the machine learning outcomes (especially strategies for providing better training data). Finally, through iterations of model building and evaluation, users learned about what was feasible to accomplish with the given features and algorithms, and they sometimes modified the learning problem definition or their goals for how the models would be used in order to create models that were most useful in practice.

From these observations, we conclude that supervised learning models intended to be ultimately used in interactive contexts should be evaluated with attention to the model qualities important to users in those contexts; in particular, objective accuracy metrics such as cross-validation may not be sufficient to allow researchers or system designers to validate or compare model quality. We also conclude that exploratory evaluation of models can complement objective metrics in allowing users to evaluate models against a wide range of criteria. We propose that IML systems may benefit from the use of existing or new algorithms designed to optimize various quantities (beyond generalization accuracy, and possibly including training accuracy) that are important to users. Lastly, we conclude that a previously-unexplored benefit of IML is the ability for human domain experts to improve the ultimate usefulness of a trained model, through iteratively using model evaluation outcomes to inform changes to the learning problem and to the context in which the model will be used.

ACKNOWLEDGEMENTS

We are grateful to our participants for their time, feedback, and insights. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation. This work is also supported by the Kimberly and Frank H. Moss '71 Research Innovation Fund and the David A. Gardner '69 Magic Project.

REFERENCES

1. Allen, J. M., Asselin, P. K., and Foulds, R. American sign language fingerspelling recognition system. In *Proc. of the IEEE 29th Bioengineering Conference* (2003).

2. Amershi, S., Fogarty, J., Kapoor, A., and Tan, D.

Overview-based example selection in end-user interactive concept learning. In *Proc. of the ACM Symposium on User Interface Software and Technology (UIST '09)* (2009), 247–256.

3. Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. Examining multiple potential models in end-user interactive concept learning. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2010), 1357–1360.
4. Baker, K., Bhandari, A., and Thotakura, R. An interactive automatic document classification prototype. In *Proc. of the Third Workshop on Human-Computer Interaction and Information Retrieval* (2009), 30–33.
5. Begga, R., and Kamruzzaman, J. A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data. *Journal of Biomechanics 38* (2005), 401–408.
6. Bishop, C. M. *Pattern Recognition and Machine Learning*, 2nd ed. Springer, 2007.
7. Fails, J. A., and Olsen, Jr., D. R. Interactive machine learning. In *Proc. of the International Conference on Intelligent User Interfaces (IUI '03)* (2003), 39–45.
8. Fels, S. S., and Hinton, G. E. Glove-TalkII: An adaptive gesture-to-formant interface. In *Proc. of Computer Human Interaction (SIGCHI95)* (1995), 456–463.
9. Fiebrink, R. *Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance*. PhD thesis, Princeton University, Princeton, NJ, USA, January 2011.
10. Fiebrink, R., Schedel, M., and Threw, B. Constructing a personalizable gesture-recognizer infrastructure for the K-Bow. In *Proc. of the 3rd International Conference on Music and Gesture (MG3)* (2010).
11. Fiebrink, R., Trueman, D., Britt, C., Nagai, M., Kaczmarek, K., Early, M., Daniel, M. R., Hege, A., and Cook, P. R. Toward understanding human-computer interaction in composing the instrument. In *Proc. of the International Computer Music Conference* (2010).
12. Fiebrink, R., Wang, G., and Cook, P. R. Support for MIR prototyping and real-time applications in the ChucK programming language. In *Proc. of the International Conference on Music Information Retrieval* (2008).
13. Fogarty, J., Tan, D., Kapoor, A., and Winder, S. CueFlik: interactive concept learning in image search. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2008), 29–38.
14. Gohring, N. Mundie: Microsoft's research depth enabled Kinect. *PCWorld* (July 2010).
15. Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S. R. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2007), 145–154.
16. Kapoor, A., Lee, B., Tan, D., and Horvitz, E. Interactive optimization for steering machine classification. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2010), 1343–1352.
17. Lee, M., Freed, A., and Wessel, D. Neural networks for simultaneous classification and parameter estimation in musical instrument control. *Adaptive and Learning Systems 1706* (1992), 244–55.
18. Peng, L., and Gerhard, D. A Wii-based gestural interface for computer-based conducting systems. In *Proc. of the Conference on New Interfaces for Musical Expression* (2009).
19. Shilman, M., Tan, D., and Simard, P. CueTIP: A mixed-initiative interface for correcting handwriting errors. In *Proc. of the ACM Symposium on User Interface Software and Technology (UIST '06)* (2006).
20. Talbot, J., Lee, B., Kapoor, A., and Tan, D. S. EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2009), 1283–1292.
21. Valiant, L. G. A theory of the learnable. *Communications of the ACM 27*, 11 (1984), 1134–1142.
22. Vredenburg, K., Mao, J.-Y., Smith, P. W., and Carey, T. A survey of user-centered design practice. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2002), 471–478.
23. Wilson, A., and Shafer, S. XWand: UI for intelligent spaces. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2003), 545–552.
24. Young, D. Classification of common violin bowing techniques using gesture data from a playable measurement system. In *Proc. of the Conference on New Interfaces for Musical Expression* (2008).