Investigating Statistical Machine Learning as a Tool for Software Development

Kayur Patel[1], **James Fogarty**[1], **James A. Landay**[1,2], **Beverly Harrison**[2]

[1]Computer Science & Engineering
DUB Group, University of Washington
Seattle, WA 98195
{kayur, jfogarty, landay}@cs.washington.edu

[2]Intel Research Seattle
1100 NE 45th Street, 6th Floor
Seattle, WA 98105
beverly.harrison@intel.com

ABSTRACT

As statistical machine learning algorithms and techniques continue to mature, many researchers and developers see statistical machine learning not only as a topic of expert study, but also as a tool for software development. Extensive prior work has studied software development, but little prior work has studied software developers applying statistical machine learning. This paper presents interviews of eleven researchers experienced in applying statistical machine learning algorithms and techniques to human-computer interaction problems, as well as a study of ten participants working during a five-hour study to apply statistical machine learning algorithms and techniques to a realistic problem. We distill three related categories of difficulties that arise in applying statistical machine learning as a tool for software development: (1) difficulty pursuing statistical machine learning as an *iterative and exploratory process*, (2) difficulty *understanding* relationships between data and the behavior of statistical machine learning algorithms, and (3) difficulty *evaluating* the performance of statistical machine learning algorithms and techniques in the context of applications. This paper provides important new insight into these difficulties and the need for development tools that better support the application of statistical machine learning.

Author Keywords

Statistical machine learning, software development.

ACM Classification Keywords

H5.2 Information Interfaces and Presentation: User Interfaces; D2.6 Programming Environments: Integrated Environments.

INTRODUCTION AND MOTIVATION

Statistical machine learning has emerged as an important tool in the development of modern software. For example, the explosion of information available on the Web has motivated work on interfaces that better support common tasks by automatically identifying relationships within and among Web pages [11, 13]. Concern for demands on human attention imposed by computing and communication systems has prompted the examination of sensor-based statistical models of human interruptibility [7, 12]. The complexity of modern software has led to the exploration of statistical techniques for detecting bugs and anomalous behavior in deployed systems [2, 28]. Advances in low-cost sensing have prompted work on activity modeling and location-aware computing to enhance the limited input and output capabilities of mobile devices [17, 19].

Statistical machine learning algorithms and techniques remain an important topic of expert study, but the above examples have the common characteristic that they are focused on applying existing algorithms and techniques to solve a problem of interest. In this sense, there are many researchers and developers who see statistical machine learning not as a topic of study, but rather as a tool for software development. Extensive prior research has explored the general difficulties faced by software developers [14, 15, 18, 24, 25], but little work has studied the application of statistical machine learning in software development. Instead, existing statistical machine learning tools (e.g. [22, 27]) have generally been developed by and for statistical machine learning experts, whose primary interest is often in developing and evaluating new statistical machine learning algorithms and techniques. But the application of statistical machine learning is no longer limited to such experts, and so it is important to understand the difficulties that developers face when applying statistical machine learning as a tool for software development.

This paper takes a two-pronged approach to examining the difficulties that arise in applying statistical machine learning as a tool for software development. We first interview eleven researchers, each of whom has significant experience applying statistical machine learning algorithms and techniques to human-computer interaction research problems. We then study ten participants working during a five-hour study to apply statistical machine learning algorithms and techniques to a realistic problem. Our interviews and our study reveal three related categories of difficulties: (1) difficulty pursuing statistical machine learning as an *iterative and exploratory process*, (2) difficulty *understanding* relationships between data and the behavior of statistical machine learning algorithms, and (3) difficulty *evaluating* the performance of statistical machine learning algorithms and techniques in the context of applications. Analyzing detailed screen and workspace captures of the work of study participants in the context of

CHI 2008, April 5 - 10, 2008, Florence, Italy.

difficulties distilled from our interviews, this paper provides important new insight into these difficulties and discusses implications for new tools to better support the application of statistical machine learning.

RELATED WORK

Several texts provide appropriate introductions to statistical machine learning algorithms and techniques [10, 23]. One of the most well-studied areas of statistical machine learning is the learning of a function $y = f(x_1, \ldots, x_n)$. Depending on whether y is a continuous or nominal variable, this problem is known as either *regression* or *classification*. Both are considered *supervised* learning, because a set of *labels* are provided as part of training data at the time the function is learned. Variables $x_1, \ldots, x_n$ are known as *features*, and each should capture some useful aspect of the problem being modeled. Importantly, features are assumed as input to statistical machine learning algorithms, and must be provided by the developer.

Existing tools for the application of statistical machine learning provide a library of implementations of common statistical machine learning algorithms, with Weka being a well-known and widely-used example [27]. Environments like YALE [22] provide additional support for configuring experiments to compare the performance of different potential algorithms. Such tools can save a statistical machine learning expert significant implementation effort, but they provide little guidance to developers who are not already familiar with how to successfully employ the provided algorithms. The body of this paper further discusses difficulties faced by developers in using current tools, as our second study includes the use of Weka as part of solving a classification problem.

A number of systems have explored the packaging of appropriate features with statistical machine learning algorithms to ease development in particular domains. For example, Fails and Olsen developed Crayons, a tool for creating camera-based systems using pixel-level classifiers [4, 5]. Crayons uses a coloring metaphor to collect labeled training data, then learns a decision tree classifier using features based on integral images. Hartmann *et al.* developed Exemplar [9], which supports the interactive specification of sensor-based recognizers through a combination of signal filters and a dynamic time warping algorithm. Fogarty and Hudson developed Subtle [6], a system focused on the sensing capabilities of typical laptop computers that uses operator-based feature generation with wrapper-based feature selection to automatically create classifiers based on labels provided by an application. Maynes-Aminzade *et al.* present Eyepatch, a tool for developing camera-based interactions that includes support for training different types of vision-based classifiers [20].

Such tools demonstrate the potential for the application of statistical machine learning algorithms and techniques, but they generally achieve their success by highly constraining both their application domain and the approaches that a developer can take to a problem. One of the most common constraints is to limit the developer to providing training data, an approach taken by both Crayons and Subtle. Both systems package a set of features and algorithms that work well in their domains, allowing the developer to provide examples of the concept they want to model. But if the packaged features and modeling algorithm are not a good fit for the model that a developer wants to learn, such tools provide little recourse. Similarly, Exemplar allows a developer to explicitly manipulate parameters to a dynamic time warping algorithm, but provides little support for determining whether this is the appropriate algorithm for a problem and no support for experimenting with other potential algorithms. In short, systems constrained in such ways provide a low floor (a low barrier to entry) at the expense of a low ceiling (the point at which the tool's assumptions and constraints become an obstacle to addressing a problem) [21]. In contrast, this work explores support for the end-to-end development of systems based on the application of statistical machine learning algorithms and techniques. By examining the difficulties that developers encounter, we aim to inform the development of new tools that provide both a low floor and a high ceiling.

Ko *et al.* identify six learning barriers faced by novice developers [16]. Although our focus is on experienced software developers who are not experts in the application of statistical machine learning algorithms and techniques, analogous barriers arise. For example, a developer who cannot conceive of how to frame a problem as a matter of learning a function $y = f(x_1, \ldots, x_n)$ is encountering a barrier analogous to Ko *et al.*'s design barriers. Our work complements such work, examining in greater depth the unique difficulties encountered in the application of statistical machine learning.

Other areas of related work include studies of engineering and creative design processes [3, 26], the challenges of effective information visualization [1], and work in knowledge discovery and data mining [8]. We note that results in all of these areas will inform the design of new tools to better support the effective application of statistical machine learning as a tool for software development. But we also note that statistical machine learning continues to grow in importance as a tool for software development, and so tools supporting the effective application of statistical machine learning warrant careful attention. As an analogy, it is clear that user interface toolkits and studies of the development of user interface software have been critical to advancing human-computer interaction [21]. Because statistical machine learning continues to emerge as an important tool in human-computer interaction and in other fields, our work aims to enable similar long-term impact via a first set of empirical studies examining the difficulties that developers face in appplying statistical machine learning.

STUDY OVERVIEWS

As noted in our introduction, we take a two-pronged approach to examining the difficulties that arise in applying statistical machine learning. We first conduct semi-structured interviews of eleven researchers experienced in applying statistical machine learning to human-computer interaction research problems. These interviews provide high-level insight into the difficulties faced by developers.

Based on this high-level insight, we then design and conduct a laboratory think-aloud examining ten participants working during a five-hour period to apply statistical machine learning algorithms and techniques to a realistic problem. We chose this combination of approaches because the development of software based on statistical machine learning algorithms and techniques typically takes place over an extended period of time, on the order of weeks to months. Starting with interviews allows a high-level exploration of difficulties that arise in applying statistical machine learning, and our lab study then probes exactly how how those difficulties manifest as developers work on a problem. This section introduces our interviews and our study, deferring results until later sections.

Semi-Structured Interviews

We interviewed eleven researchers with significant experience applying statistical machine learning algorithms and techniques to human-computer interaction problems. To avoid confusion when discussing our two groups of participants, this paper refers to our interview participants as IP1 through IP11. As an indication of the breadth of their experience, we note that these researchers have worked on such problems as intelligent digital photo management, vision-based facial expression recognition, availability modeling in instant messaging, EEG-based recognition of brain activity, RFID-based activity recognition for elder care applications, accelerometer-based activity recognition for fitness applications, mixed-initiative pen-based text input, programming-by-demonstration approaches to text editing, interactive tools for creating camera-based interfaces, automated network packet diagnosis, and models of musical style. Each interview participant has published multiple papers in top venues. Participants were selected to include a mixture of backgrounds, including researchers from the statistical machine learning community who are focused on human-computer interaction applications and researchers from the human-computer interaction community who have incorporated statistical machine learning in their work.

Each participant recalled two to three prior projects that included the application of statistical machine learning algorithms and techniques. They then described the lifecycle of the project, from conception to completion, as well as how the application of statistical machine learning related to other aspects of the project. We asked participants to diagram their process while describing the project, and we elicited further discussion and clarification by concurrently annotating and editing their diagrams. After discussing this first project, participants compared and contrasted it with the other projects they had initially discussed. Interviews lasted for between 40 and 90 minutes, and we captured audio recordings for later transcription and review.

Interview Results Overview

Although we defer the bulk of discussion until after the presentation of our think-aloud study, several results from our interviews directly inform the design of our think-aloud study. First, our interviews revealed that the application of statistical machine learning is a highly *iterative and exploratory process*. A typical process requires the formulation of a learning problem, collection of appropriate training data, the extraction of features from the data, the selection of a modeling algorithm, and experimentation to determine whether the resulting system meets the needs of the application. Although these steps describe a set of linear dependencies, our participants emphasized the fact that the actual development of a system is much more exploratory than such linear dependencies suggest. This led us to ensure that our think-aloud study examined this entire process (as opposed to, for example, focusing only on developer model selection given a predetermined set of features). Second, our interviews revealed the importance of *understanding* relationships between data and the behavior of statistical machine learning algorithms in order to decide how to proceed. We therefore desired a modeling problem that can be effectively solved using many different approaches, as opposed to one that forced participants towards a single effective solution. Third, our interviews revealed difficulties with *evaluating* the performance of statistical machine learning in the context of applications, as our interview participants felt that they must often manage concerns other than the straightforward notion of model accuracy. We therefore designed our think-aloud study to examine two other concerns that interview participants raised, a need for systems to work well when used by different people and a need to consider computational cost and implications for responsiveness when developing interactive applications.

Think-Aloud Study

Based on the difficulties described by our interview participants, we designed the *Digits* task to examine how these difficulties manifest as developers work on a problem. This subsection presents the task, the development environment used by participants, our experimental procedure, and our participants.

The Digits Task

Illustrated in Figure 1, the Digits task has participants create a image-based classifier of handwritten digits, recognizing digits between 0 and 9. We chose this task after experimenting with several possibilities, finding that the task is easy enough for participants to produce a reasonable classifier within a matter of hours, but hard enough that participants employ a variety of strategies and create classifiers of varying performance. Regarding our just-discussed second requirement for a task (that it can be effectively solved using many different approaches), we note both that image-based handwritten digit recognition is a well-studied problem with many known approaches. Good

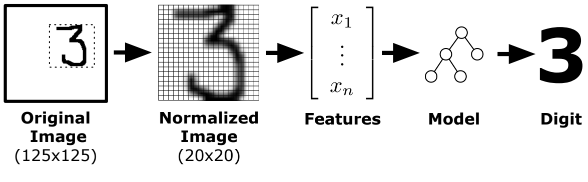


Figure 1. Our Digits task requires recognition of handwritten digits. Participants collect data, extract features from the images, apply statistical machine learning algorithms, and evaluate the effectiveness of their system.

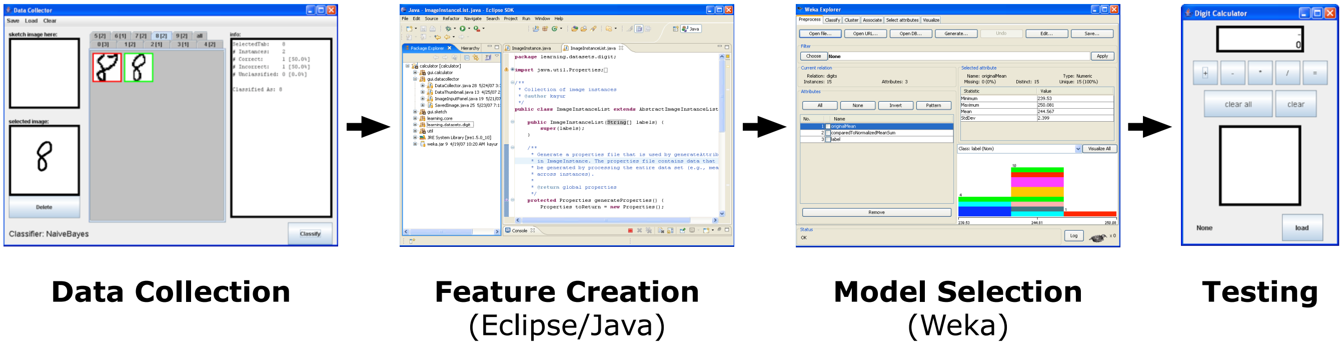


Figure 2. The toolset used by participants in our think-aloud study.

results were obtained by participants who placed varying degrees of emphasis upon data collection, feature creation, and model exploration. Finally, the use of an image-based task provides a natural visual representation of data, and early experimentation with other tasks indicated this was critical to tasks being tractable in the time allotted.

Development Environment

Participants worked using a toolset, illustrated in Figure 2, that included the entire process of collecting training data, extracting features from the images, choosing a modeling algorithm, and then interactively testing their model in a simple application. This toolset is typical of the tools that our interview participants discussed using in their work.

The first component is a data collection application. Participants used this application with a stylus and a 1024x768 Wacom Cyntiq tablet display. By selecting the tab corresponding to a digit between 0 and 9, participants could create labeled instances of that digit. Digits were captured as 125x125 monochrome images and as normalized 20x20 grayscale images (see Figure 1). Participants could create and manipulate multiple data files containing such images, and could view how a model classified the images in a file.

The second component is the Eclipse development environment for Java, used to extract features from the digit images. Existing statistical machine learning tools (including Weka, discussed next) generally parse feature values from simple tabular file formats. Our interview participants reported using typical software development tools to create small programs that parse their raw data (images in the Digits task), compute feature values, and store those feature values in a format appropriate for their statistical machine learning tool. Because our focus is the application of statistical machine learning, and because the development of such parsers can be an error-prone and time-consuming process, we provided participants with a codebase that parsed the file format used by the data collection tool, provided stub methods to compute features based on the original 125x125 monochrome images and the normalized 20x20 grayscale images, then stored the resulting feature values in an appropriate file format. As a part of introducing the study, participants were shown where and how to edit the feature creation stubs.

The third component is Weka, a well-known and widely-used statistical machine learning tool [27]. Weka provides a large library of feature visualizations, filters, and modeling algorithms. Participants used Weka to load the tabular feature files output by their feature creation code, to apply potential modeling algorithms, and to evaluate model accuracy using standard evaluation techniques (such as ten-fold cross-validations).

The last component is a simple interactive calculator application used for testing the system. This application used the participant's feature creation code and a Weka-exported model to interactively classify pen input. We included this for two reasons. First, many of our interview participants reported that "trying it out" was an important part of their process of applying statistical machine learning. Providing this application closed the loop and allowed participants to interact with the model they created. Second, the interactive use of a model exposes important aspects of that model that are not captured when only considering accuracy, most notably the computational cost of the features used by the model. Participants were told their models needed to work well in an interactive context, and if the features or the algorithm used by a participant were too computationally expensive, a noticeable lag would result.

Procedure

Participants worked in a small office free of distractions, and they were asked to think aloud as they worked. Links to the Java API and the Weka API were provided, and participants were free to use any resources they felt would be helpful. Nearly all chose to use the Web to find information about features, and several downloaded code to compute features. The workstation included a 24" Dell 2407WFP display running at 1900x1280 and the Wacom Cyntiq tablet display running at 1024x768. Participants were free to use the available monitor space however they chose. Most used the primary display for interacting with the Eclipse development environment and with Weka, thus using the tablet display almost exclusively for interacting with the pen-based data collection and calculator applications. Commercial screen capture software continuously captured the desktop, a video camera recorded the physical environment, and custom software took continuous snapshots of the files in participant workspaces. To minimize the impact of the computational demands of statistical machine learning tools and our capture software, participants worked alone on a computer with two quad-core Xeon processors (at 2.66 GHz) and four gigabytes of RAM.

Each session started with a tutorial, familiarizing participants with the toolset by stepping them through the collection of a handful of labeled digit images, the extraction of a simple feature, the creation of a simple model, and the interactive use of that model in the calculator application. They were then given two hours to collect data, develop features, and create the best classifier they could. After a break, we provided participants with 200 labeled digit images collected from four different people (5 examples of each digit per person). We provided this data to see how participants would use it in developing their system. Participants then had another two hours to continue developing their system, using both the data we provided and data they collected. Participants were told that their classifier would be evaluated according to its accuracy for digits sketched by other people (subject to the constraint that it worked fast enough for interactive use). Participants received a $50 gift certificate for participation, and the participant who created the best model received an additional $50 gift certificate.

Participants

We recruited ten participants, all computer science graduate students. To avoid confusion when discussing our two groups of participants, this paper refers to our think-aloud participants as TAP1 through TAP10. We chose this population after early experimentation showed that people with no prior exposure to statistical machine learning algorithms and techniques could make very little progress within such a short period of time. All of our participants therefore had previous classroom exposure to statistical machine learning algorithms and techniques. They also reported prior experience developing in Java and using the Eclipse IDE. Four out of the ten had previously used Weka, and three of those four had worked with Weka's API. This population is consistent with the goals of our work, as we are not focused on novice software developers. We are instead focused on people who already have some idea of how to apply statistical machine learning, but are not necessarily highly-trained experts in statistical machine learning.

Think-Aloud Results Overview

Figure 3 presents the final accuracy of each think-aloud participant's system, as well as a plot of the evolution of the accuracy of each participant's system over the course of each of the two-hour portions of the study. These accuracies have been computed using 2000 labeled digits collected from 20 different people (10 examples of each digit per person), none of which were provided to any of the think-aloud participants. The plots were computed by scripts that created and tested models based on our automatically captured continuous snapshots of participant workspaces. TAP2's system made heavy use of a set of files outside the environment we captured, and so we are unable to plot the accuracy of TAP2's models over time.

MACHINE LEARNING AS AN EXPLORATORY PROCESS

A number of dependencies exist when developing an application based on learning a function $y = f(x_1, \ldots, x_n)$. A developer must define a formulation of y that is appropriate for their application. Training data must be collected and labeled according to y. Features $x_1, \ldots, x_n$ need to be defined and extracted from the raw data. Each of these depends on the previous, and a modeling algorithm can be applied to learning $y = f(x_1, \ldots, x_n)$ only after all of these dependencies have been met.

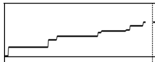
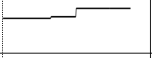
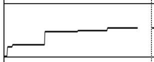
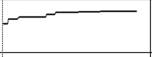
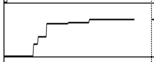
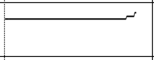
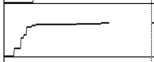
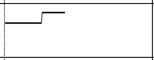
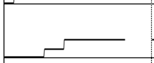
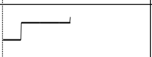
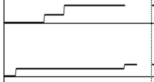
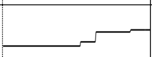
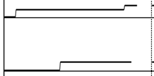
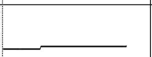
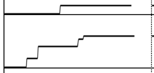
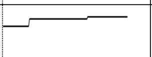
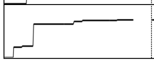
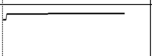
	First 2 Hours	Second 2 Hours	Final Accuracy
TAP1			84.7%
TAP2	X	X	75.3%
TAP3			78.3%
TAP4			82.9%
TAP5			84.7%
TAP6			78.0%
TAP7			56.9%
TAP8			22.8%
TAP9			78.8%
TAP10			84.4%

Figure 3. The final accuracy of each think-aloud participant's model, as well as a plot of the accuracy of each participant's model over the course of each two-hour session.

While this linear set of dependencies might suggest a linear process in the application of statistical machine learning algorithms and techniques, our interviews revealed a fundamentally exploratory and iterative process. In discussing and diagramming their previous projects, interview participants emphasized the non-linearity of their work and described situations where an apparent dead end for a project was overcome by revisiting an earlier point in their process. For example, IP6 described a long and fruitless exploration of modeling algorithms in collaboration with expert statistical machine learning colleagues. IP6's breakthrough came when they went back and questioned whether their features were appropriate, and they then found that the creation of new features led to good results with a simple modeling algorithm. In their words, "We basically tried a whole bunch of Weka experimentation and different algorithms ... and nothing worked, so we decided that ... maybe we should explore the feature space." In another example, IP5 described a frustration with the fact that a set of features and a modeling algorithm worked well on training data but poorly when deployed in an application. In this case, their breakthrough was in realizing that their system had latched onto the characteristics of a single person who had provided the vast majority of their training data. The collection of more training data led to significantly improved results. As a final example, IP5 recounted a case where they changed the definition of the learning problem based on the performance of their system. A model was generally performing well at recognizing a set of activities, but often confused two related activities. Within the context

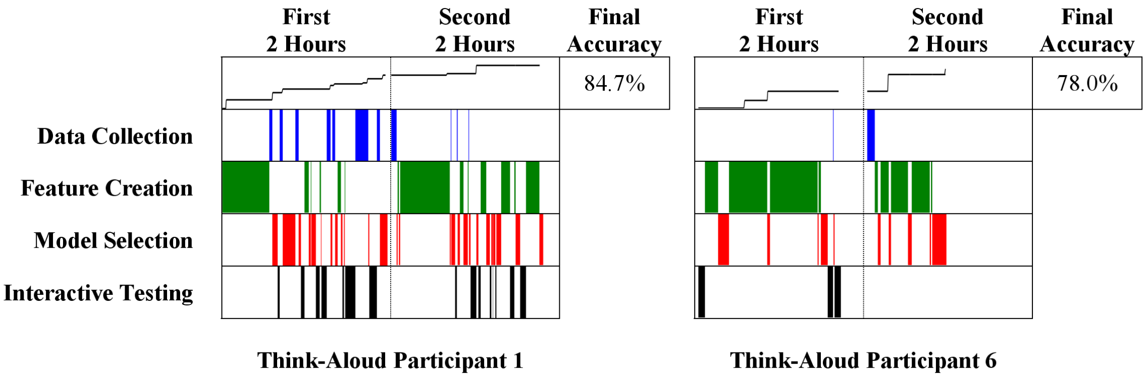


Figure 4. A comparison of the processes of two representative think-aloud participants. TAP1 makes steady progress by iteratively exploring all portions of the problem, while TAP6 spends that first portion of the study overly focused on feature creation.

of the desired application, IP5 decided it was more important to reliably recognize a set of activities than it was to distinguish between the two particular activities that were being confused. They therefore combined the two activities into a single activity, modifying their application and their definition of y to balance the needs of their application with the performance of their inference components.

Our think-aloud study also revealed the importance of exploratory and iterative development. Figure 4 contrasts the process of two representative think-aloud participants. TAP1's continuous exploration of every component of the modeling process yields steady progress, while TAP6 struggles in part because they become overly focused on a subset of the modeling process. This is illustrated via our labeling of what activities these participants were performing at each point in the task, based on a manual coding of our collected screen capture recordings. TAP1 starts by investing some effort into feature creation, then begins a period of iteratively creating new training data, revising their feature creation code, considering different modeling algorithms, and testing the resulting system. This iterative exploration can be seen in the very dashed nature of TAP1's activities, as the participant quickly moves through all portions of the problem. In contrast, TAP6 spends almost the entire first half of the task focused on feature creation and makes relatively little progress, creating a model with one of the lowest midpoint accuracies. Once provided with additional training data (the 200 examples from four people that we provided each think-aloud participant at the midpoint of the task), TAP6's model is noticeably improved (and screen capture recordings show that TAP6 made no significant changes to their features or their modeling algorithm). TAP6 was unable to make this progress earlier at least in part because they were overly focused on feature creation and did not revisit their training data. TA1's final model is a top performer, while TAP6's final model performs poorly in comparison to other models.

Given these results, we note that our interview participants from the statistical machine learning community generally prefer to use a tool like Matlab to implement their entire end-to-end system, often coding their own algorithms. The decision to implement their own algorithms in a tool like Matlab may at first seem like a simple matter of personal preference, but our interview participants describe it as important to determining how to best proceed in addressing a problem. For example, I10 stated "I think it's really valuable to work in an interactive environment, [because] you can go back and ask a data structure 'what did you do?' or you can add three lines and save off the state in some way." Similarly, I10 said "If you have a black box that is a [statistical machine learning algorithm] and it produces numbers in the end, then you have no idea what actually happened. So you need to be able to look inside the state of the algorithm and see what is happening, just like you would a program."

Commonly-used tools introduce gaps into the process of applying statistical statistical machine learning in software development. The toolset used in our think-aloud study (recall Figure 2) is typical of commonly-used toolsets, as Weka allows the application of statistical machine learning algorithms without a need to implement or understand the details of those algorithms. This is valuable, but comes at the expense that gaps between tools impede exploratory and iterative development.

One example of such a gap occurs when features $x_1, \ldots, x_n$ are computed from raw data (generally by small custom programs). When these features are loaded in a tool for applying statistical machine learning algorithms, there is typically no way to link the set of features to the raw data from which they were computed. For example, participants in our think-aloud study loaded the output of their feature creation code in Weka, but once within Weka they were unable to see the actual image of the digit associated with each set of values $x_1, \ldots, x_n$. This gap is a barrier to debugging a system, and is typical of existing tools that have chosen to be agnostic to how feature values were obtained because they package only a portion of the exploratory and iterative process of applying statistical machine learning algorithms and techniques. Given this gap, our discussion addresses the potential for tools that support the entire process of applying statistical machine learning algorithms and techniques.

A second gap introduced by current tools that are agnostic to how feature values are obtained is the need for developers to implement or obtain implementations of commonly used features. Our interview participants reported that they often start their feature creation process by investigating what features have worked well for other people with similar data. Our think-aloud participants also used the Web to investigate features that work well for problems similar to our Digits task, with varying success. For example, TAP4 and TAP10 found discussions of the same feature when reviewing related work. TAP4 was able to find an implementation of that feature and successfully incorporate it into their system, while TAP10 spent time looking for implementations but eventually abandoned the feature. Informed by this gap, our discussion addresses the potential for new tools to package and support the reuse of common feature computations.

The final gap we discuss here is compounded over time as a developer explores a problem. As developers collect new data, explore new features, and consider different modeling algorithms, a variety of intermediate artifacts are created. These include raw data files, files containing the results of various feature computations (including variations on what raw data was used and what code was used to compute features from the raw data), and different models. But this experimental history is not supported by current toolsets, and we observed our think-aloud participants using fragile strategies to maintain a history of their experimentation. For example, TAP1, TAP5, TAP9, and TAP10 kept paper logs of configurations they had tried and the accuracy they had obtained with those approaches (interestingly, these four participants also produced some of the best models). Participants also used filenames like "LogitBoostWith8To18EvenWindow-Iter=10.model" to attempt to document their exploration, though we note that such a strategy is likely only effective in the short term (the filename does not, for example, indicate what data was used to train the model or what features were computed for use in the model). Our discussion therefore discusses the potential for new tools to explicitly support experimentation and record keeping.

UNDERSTANDING DATA AND ALGORITHMS

Our interview participants noted that many people who are new to the application of statistical machine learning algorithms and techniques (sometimes including the participants themselves at earlier points in their career) want to treat statistical machine learning algorithms as black boxes. Although no experienced software developer would expect a sorting algorithm to be effective without being told how to compare two objects, developers who are new to the application of statistical machine learning sometimes expect that it will somehow "just work." This section discusses four examples of the role of understanding in the application of statistical machine learning algorithms and techniques.

IP2 has an extensive statistical machine learning background and significant experience applying their background to human-computer interaction problems. In describing their general approach, they noted they initially focus on creating promising features. IP2 reported that they understand the workings of statistical machine learning algorithms to the point that they can examine simple visualizations of feature values and determine whether an algorithm is going to work well. After collecting appropriate data, IP2 iteratively creates and examines features. They generally do not apply a learning algorithm until after they are confident their features will work well. This approach represents one extreme, as the participant feels that they understand modeling algorithms to the point that they can simply inspect a set of features and know whether they will work well in a model.

A second process emphasizing understanding and features was described by IP1, a participant with a human-computer interaction research background. This participant indicated that they are not practiced in the application of statistical machine learning algorithms, and so they instead often take an approach based on manual specification of heuristics. They described a process based in collecting data and creating features (just like our other participants), but then relating those features to desired behaviors using manually-coded rules and manually-determined thresholds. IP1 prefers this approach because, in the case where a system does not perform as expected, they can manually examine their features and step through their heuristics to determine why the system failed. They can then address the failure by adding a new feature or a new heuristic. So while IP1 felt that the heuristics they create might not perform as well as a system would if it were developed using a modeling algorithm, they preferred the heuristics because they enabled a greater level of understanding and ability to debug.

We saw a similar use of simple algorithms for the sake of understandability in the processes employed by some of our think-aloud study participants. For example, TAP9 initially used a decision tree algorithm because it allowed TAP9 to easily see what features were being used and what relationships existed between features at different levels in a tree. Later in the study, after TAP9 was no longer looking to create new features, they transitioned to using more complex models in search of increased performance. Their final model (a boosted ensemble of support vector machines) did indeed perform better than a decision tree, but would be near impossible to manually inspect.

The previous examples demonstrate the importance of understanding in the case where a system does not yet appear to work correctly, but understanding is important even after a system seems to work well. IP3 recounted a case where they spent several months believing they had an effective solution to a modeling problem that was examining analyses of online forums. Another researcher had provided a data file containing values for a number of features computed from forum posts, and IP3 had developed a model that performed well when tested using standard evaluation methodologies (specifically, using cross-validation techniques). But IP3 then discovered that much of the reliability of the model was based on a single feature, and that this feature did not appear to be related to the concept they were modeling. Further investigation revealed the forum had been heavily spammed during the time when the dataset was being collected, and that all of the spam messages were of the same class in IP3's formulation of the modeling problem. IP3 had therefore accidentally built a spam detector. Although the model did perform well at detecting spam forum posts, it was not solving the problem that IP3 intended to solve.

In the first three examples discussed in this section, participants employed approaches based in understanding how data relates to a model. Though the exact processes of IP2, IP1, and TAP9 vary significantly, they all took approaches based in building up an understanding of their data, their features, and how models interact with those features. Our final example illustrated the risks of treating portions of the process as a black box, as the dataset and features underlying IP3's work led to models that were capturing a concept very different from what IP3 thought they were capturing. Our interview participants from the statistical machine learning community commented on the tendency of people who are new to the application of statistical machine learning to want to treat algorithms as black boxes. IP10, for example, warns that statistical machine learning algorithms cannot do "semantic things" but that successful classification instead requires "really seeing what is happening with the data." IP4 similarly cautions that if a person cannot understand at some level the differences between data, then it will be difficult or impossible to build a classifier that can. In contrast to propagating the misconception that statistical machine learning algorithms can be treated as black boxes, tools need to support the role of the developer in exploring and solving a problem. Our discussion comments on the potential for new tools to emphasize visualizations of models, data, and their relationships.

EVALUATING PERFORMANCE IN APPLICATIONS

In addition to difficulties creating systems based on the application of statistical machine learning algorithms and techniques, our interview participants discussed difficulties evaluating the performance of such systems in the context of applications. The statistical machine learning community has developed a number of techniques for evaluating the accuracy of models, but our interview participants felt that they must often manage concerns outside the focus of traditional evaluation metrics and methodologies.

One recurring difficulty is due to differences in data collected from different people and a need to understand how a system is likely to perform when deployed with people other than those who provided the system's training data. At the core of this difficulty is the IID assumption, that data is independently and identically distributed. Many commonly used statistical machine learning algorithms and techniques make this assumption, and the fact that different people may wear a sensor differently or draw a digit differently violates the assumption. Although this is not always problematic, as statistical machine learning algorithms and techniques often work well in spite of such violations of their theoretical basis, our interview participants reported a number of related difficulties. For example, IP5 recalled that "the cross-validation would show ... 85% to 90% accuracy .. and then you would try it ... it worked extremely well for some people and not well for others." In contrast to developing a single model to work well across people, IP4 reported a strategy of creating multiple models. At the beginning of a deployment, IP4 then tested which model seemed to be the best fit for each person in the deployment.

	Participant	2000 Digit		Number of
	Test	Test	Accuracy	Training
	Accuracy	Accuracy	Test Error	Examples
TAP1	87.8%	84.7%	3.1%	688
TAP2	98.0%	75.3%	22.7%	200
TAP3	89.1%	78.3%	10.8%	258
TAP4	95.6%	82.9%	12.7%	250
TAP5	91.3%	84.7%	6.6%	425
TAP6	90.4%	78.0%	12.4%	230
TAP7	72.0%	56.9%	15.1%	200
TAP8	26.5%	22.8%	3.7%	200
TAP9	92.8%	78.8%	14.0%	320
TAP10	93.4%	84.4%	9.0%	500

Figure 5. A comparison of how well participant's own tests indicated their models performed, how well they performed on 2000 new test digits, and how many training examples each participant used.

Our Digits task included providing participants with training data from four different people so that we could examine how participants used that data in constructing and evaluating their models. Participants might, for example, have trained their system using their own data and tested it against the provided data. Or they might have conducted cross-validations that trained a model using data from three of the people, then tested against the data from a fourth. Either of these approaches would have likely provided some insight into how well their system will work with new people. Instead, all participants trained a single model using data from all four people (often adding data of their own), then evaluated it using randomized 10-fold cross-validation. While randomized cross-validation is a standard technique for testing a system given a dataset for which the IID assumption holds, it ignores the fact that the data was collected from four people. TAP10 initially began to take an approach based in training with data from three people and testing against a fourth, but instead used randomized cross-validation because it was easier within Weka.

The left side of Figure 5 shows one consequence of think-aloud participants using cross-validation methodologies that ignored the relationship among data collected from the same person. In comparison to evaluations that the participants performed, every model performs worse when tested against our final set of 2000 digits. A paired t-test indicates that participant estimates of how well their models performed are significantly higher than performance measures obtained using our 2000 test digits ($t(9) = 5.96, p < .001$). Such discrepancies can significantly impact a developer's process: TAP2, for example, quit the task with time remaining because their evaluations showed their model performing at 98.0% accuracy, though its performance in our tests was much lower.

The most common strategy for addressing concerns related to the IID assumption (or other data quality concerns) is to collect large enough of a dataset that the concern no longer applies. Indeed, state-of-the-art systems for recognition problems like that in our Digits task are generally based in using massive training datasets. Evidence of this can also be seen by examining how many example instances each of our think-aloud participants used in relation to how closely their estimates of model performance matched our final tests (see the right side of Figure 5). An analysis of variance, excluding TAP8 (whose failure to produce an effective model makes them an outlier for this analysis), shows that the number of training examples each think-aloud participant used had a significant effect on how well their estimates of model performance corresponded to tests performed using our 2000 test digits ($F(1,7) = 14.00, p < .01$). While the collection of large datasets is a powerful approach, our interview participants noted that the cost of such data collection can be prohibitive. For example, when asked whether they collected more data to further explore a problem, IP10 responded "No. It was way too hard. There was no question." Collecting more data is not always an option, our discussion considers new approaches to tools that help developers examine common data quality concerns.

In addition to concerns about how to effectively evaluate model accuracy, our interview participants noted that the application of statistical machine learning requires addressing concerns other than accuracy. In discussing modeling based on personal activity histories, IP11 discussed a need to balance the potential utility of a feature against the privacy implications of collecting the data that would be needed to compute the feature, saying "this was kind of a tradeoff between what we would have wanted to have and what we can have." IP8 discussed the computational cost of features, noting "If your document's large, then it takes a lot of time." and "Part of [making the algorithm faster] was to cut back on the features." In our think-aloud study, TAP2 faced a similar tradeoff, as they used an algorithm to automatically generate a large number of features and their full set of features was too computationally expensive for interactive use. They therefore used a feature selection algorithm to reduce the number of features used by their algorithm. Interestingly, their decision to use a feature selection algorithm based on randomized 10-fold cross-validation using their entire dataset (as opposed to configuring a feature selection process to find features that work well across different people) probably led to significant overfitting and hurt their model's performance when tested against our 2000 new test digits. Our discussion considers potential new approaches to tools to help developers address such concerns.

IMPLICATIONS AND DISCUSSION

First, it is clear that non-expert tools need to support the entire exploratory and iterative process of applying statistical machine learning algorithms. Existing tools that focus exclusively on model selection (assuming features as input, typically in a tabular file format) introduce gaps into a process that requires concurrent exploration of data, features, and models. Once data collection and feature creation are embraced in an integrated tool for the application of statistical machine learning algorithms and techniques, a number of possibilities emerge. Our interview participants reported that they often start by finding what features have previously worked well in a domain, and most of our think-aloud participants used Web search engines to try to find features that work well for problems like our Digits task. Integrated tools could provide libraries of feature computations for well-studied domains, perhaps employing a community database to enable the sharing of new feature implementations. Beyond simply providing a library of feature computations, integrated tools could use automated analyses of a developer's dataset to identify features that seem appropriate, suggesting them to the developer. Supporting the entire process also enables support for automated record keeping and experimentation. Instead of relying upon brittle strategies like embedding the parameters of an experiment in the filenames of logs, developers could focus on exploring their problem while knowing that the tool will allow them to revisit and compare different approaches they have explored.

Second, the successful application of statistical machine learning algorithms and techniques requires understanding, as opposed to the expectation that algorithms can be treated as black boxes. An integrated tool should provide not only visualizations of data, features, and models, but also visualizations of their relationships. Whereas existing tools generally output simple textual presentations of a model evaluation (such as confusion matrices or accuracy measures), an integrated tool should allow a developer to see *what* data was misclassified and *how* the misclassification occurred. For example, an integrated tool could support the examination of a misclassification by showing what features were branched upon within a decision tree or what data it was compared to in a nearest-neighbor algorithm.

Third, developers applying statistical machine learning algorithms and techniques are ultimately focused on the success of their application. An integrated development tool should therefore support a rapid and lightweight transition from exploring the creation of a model to deploying that model in an application, perhaps through automated support for exporting a model and the feature computations it relies upon in an executable format. This is in contrast to the current need for developers to decide how to migrate the ad-hoc and often inefficient feature computation scripts they developed during model exploration. Although randomized cross-validations are attractive when data meets the IID assumption, tools also need to explicitly support evaluations related to common concerns in realistic data (such as relationships between data collected from the same person). An integrated tool might even attempt to detect the occurrence of such concerns in a dataset, confirming its findings with a developer and suggesting appropriate actions. Finally, tools should incorporate methods for developers to consider aspects of features other than their contribution to the accuracy of a model, such as a feature's computational costs or the privacy tradeoffs it implies.

Machine learning algorithms and techniques are increasingly important, and this paper examines obstacles to developer application of statistical machine learning. We are focused on experienced software developers who see statistical machine learning not as a topic of study, but rather as a tool for software development. Our interviews and our think-aloud study reveal three related categories of obstacles: (1) difficulty pursuing statistical machine learning as an *iterative and exploratory process*, (2) difficulty *understanding* relationships between data and the behavior of statistical machine learning algorithms, and (3) difficulty *evaluating* the performance of statistical machine learning algorithms and techniques in the context of applications. By identifying and examining these obstacles, this paper provides an important foundation for tools to enable the application of statistical statistical machine learning algorithms and techniques as a part of a software development process.

ACKNOWLEDGEMENTS

We would like to thank both the interviewees and the study participants, as well as the many other people who have contributed to this work. Special thanks to Eytan Adar, Krzysztof Gajos, Miryung Kim, Parag Singla, and Dan Weld. This work was supported in part by a gift from Intel Research, by an equipment donation from Intel's Higher Education Program, and by Kayur Patel's NDSEG Fellowship.

REFERENCES

1. Card, S.K., Mackinlay, J.D., Shneiderman. B. *Information Visualization*, 1999. Morgan-Kaufmann.
2. Chen, M.Y., Kiciman, E., Fratkin, E., Fox, A., and Brewer, E. Pinpoint: Problem Determination in Large, Dynamic Internet Services. *International Conference on Dependable Systems and Networks* (ICDSN 2002), 595–604.
3. Cross, N. Descriptive Models of Creative Design. *Design Studies*, **18**(4) (1997), 427–440.
4. Fails, J.A. and Olsen, D. A Design Tool for Camera-Based Interaction. *ACM Conference on Human Factors in Computing Systems* (CHI 2003), 449–456.
5. Fails, J.A. and Olsen, D. Interactive Machine Learning. *International Conference on Intelligent User Interfaces* (IUI 2003), 39–45.
6. Fogarty, J. and Hudson, S.E. Toolkit Support for Developing and Deploying Sensor-Based Statistical Models of Human Situations. *ACM Conference on Human Factors in Computing Systems* (CHI 2007), 135–144.
7. Fogarty, J., Ko, A.J., Aung, H.H., Golden, E., Tang, K.P., and Hudson, S.E. Examining Task Engagement in Sensor-Based Statistical Models of Human Interruptibility. *ACM Conference on Human Factors in Computing Systems* (CHI 2005), 331–340.
8. Hand, D. Data Mining: Statistics and More? *The American Statistician*, **52**(2) (1998), 112–118.
9. Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S.R. Authoring Sensor Based Interactions Through Direct Manipulation and Pattern Matching. *ACM Conference on Human Factors in Computing Systems* (CHI 2007), 145–154.
10. Hastie, T., Tibshirani, R., and Friedman, J.H. *The Elements of Statistical Learning*, (2001). Springer.
11. Hoffmann, R., Fogarty, J., and Weld, D.S. Assieme: Finding and Leveraging Implicit References in a Web Search Interface for Programmers. *ACM Symposium on User Interface Software and Technology* (UIST 2007), 13–22.
12. Horvitz, E. and Apacible, J. Learning and Reasoning About Interruption. *International Conference on Multimodal Interfaces* (ICMI 2003), 20–27.
13. Huynh, D.F., Miller, R.C., and Karger, D.R. Enabling Web Browsers to Augment Web Sites' Filtering and Sorting Functionalities. *ACM Symposium on User Interface Software and Technology* (UIST 2006), 125–134.
14. Kim, M., Bergman, L., Lau, T., and Notkin, D. An Ethnographic Study of Copy and Paste Programming Practices in OOPL. *International Conference on Software Engineering* (ICSE 2004), 83–92.
15. Ko, A.J., Aung, H., and Myers, B.A. Eliciting Design Requirements for Maintenance-Oriented Ides: a Detailed Study of Corrective and Perfective Maintenance Tasks. *International Conference on Software Engineering* (ICSE 2005), 126–135.
16. Ko, A.J., Myers, B.A., and Aung, H.H. Six Learning Barriers in End-User Programming Systems. *Symposium on Visual Languages and Human Centric Computing* (VLHCC 2004), 199–206.
17. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I.E., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., and Schilit, B.N. Place Lab: Device Positioning Using Radio Beacons in the Wild. *International Conference on Pervasive Computing* (PERVASIVE 2005), 116–133.
18. LaToza, T.D., Venolia, G., and DeLine, R. Maintaining Mental Models: A Study of Developer Work Habits. *International Conference on Software Engineering* (ICSE 2005), 492–501.
19. Lester, J., Choudhury, T., Kern, N., Borriello, G., and Hannaford, B. A Hybrid Discriminative/Generative Approach for Modeling Human Activities. *International Joint Conference on Artificial Intelligence* (IJCAI 2005), 766–772.
20. Maynes-Aminzade, D., Winograd, T., and Igarashi, T. Eyepatch: Prototyping Camera-Based Interaction Through Examples. *ACM Symposium on User Interface Software and Technology* (UIST 2007), 33–42.
21. Myers, B., Hudson, S. E., and Pausch, R. 2000. Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, **7**(1) (2000), 3–28.
22. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. Yale: Rapid Prototyping for Complex Data Mining Tasks. International Conference on Knowledge Discovery and Data Mining (KDD 2006), 935–940.
23. Mitchell, T.M. *Machine Learning*, (1997). McGraw-Hill.
24. Robillard, M.P. and Coelho, W. How Effective Developers Investigate Source Code: An Exploratory Study. *IEEE Transactions on Software Engineering*, **30**(12) (2004), 889–903.
25. Rosson, M.B. and Carroll, J.M. The Reuse of Uses in Smalltalk Programming. *ACM Transactions on Computer-Human Interaction*, **3**(3) (1996), 219–253.
26. Smith, R.P. and Tjandra, P. Experimental Observation of Iteration in Engineering Design. *Journal of Research in Engineering Design*, **10**(2) (1998), 107–117.
27. Witten, I.H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, (2005). Morgan Kaufmann.
28. Zheng, A.X., Jordan, M.I., Liblit, B., Naik, M., and Aiken, A. Statistical Debugging: Simultaneous Identification of Multiple Bugs. *International Conference on Machine Learning* (ICML 2006), 1105–1112.