Examining Difficulties Software Developers Encounter in the Adoption of Statistical Machine Learning

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

Statistical machine learning continues to show promise as a tool for addressing complex problems in a variety of domains. An increasing number of developers are therefore looking to use statistical machine learning algorithms within applications. We have conducted two initial studies examining the difficulties that developers encounter when creating a statistical machine learning component of a larger application. We first interviewed researchers with experience integrating statistical machine learning into applications. We then sought to directly observe and quantify some of the behavior described in our interviews using a laboratory study of developers attempting to build a simple application that uses statistical machine learning. This paper presents the difficulties we observed in our studies, discusses current challenges to developer adoption of statistical machine learning, and proposes potential approaches to better supporting developers creating statistical machine learning components of applications.

Introduction

Statistical machine learning (ML) algorithms have been used to address important problems in a wide variety of domains, including adaptive user interfaces, autonomous robotics, and protein folding. While there are already examples of applications that benefit from components that apply statistical ML, most software developers lack the specialized knowledge needed to effectively apply statistical ML algorithms and techniques. Although many developers see the value of statistical ML and want to apply it within their applications, in practice they often find it difficult to effectively apply statistical ML.

The human-computer interaction (HCI) community has extensively studied developer adoption of tools and techniques. One theme of that research is that developers often follow a *path of least resistance*, and so successful tools naturally lead developers towards doing the right things (Myers, Hudson, and Pausch 2000). Given the increasing importance of statistical ML, we note two motivations for investigating new tool support. First, many developers still opt to use hand-coded heuristics, or otherwise scale back functionality, instead of integrating statistical ML into an application. Second, many developers pursue ineffective (or even inappropriate) strategies when using current tools, as those tools do not naturally lead to developers to effective strategies.

If current advances in statistical ML are to be more widely adopted by software developers in the larger context of applications, both motivations need to be addressed. Tools need to *enable* the effective application of statistical ML, providing the support that developers need when integrating statistical ML into their applications. But tools also need to *encourage* developers to pursue effective strategies. Beyond just providing the ability to configure experiments, for example, tools also need to help developers ensure that they are conducting the right experiments. Building such tools requires an understanding of current best practices in statistical ML, but also requires an understanding of the difficulties that developers encounter when they attempt to apply statistical ML.

As a first step towards new tools, we interviewed researchers with expertise in using statistical ML to build applications. Based on our interviews, we next designed a laboratory study in which developers built the statistical ML component of a simple application. This allowed us to observe the process of applying statistical ML with a focus on the specific difficulties discussed by our interview participants. Based upon our full results (Patel et al. 2008), this paper discusses the difficulties we identified and their implications for building tools, designing algorithms, and presenting statistical ML to developers.

Related Work

Extensive prior work explores the difficulties that software developers encounter (Ko, Myers, and Aung 2004). However, relatively little work has studied the use of statistical ML as a tool for software development. Although the research community has reflected on this problem (Hand 1998), we are not aware of work that explicitly studies developers and their experiences applying statistical ML. Our studies begin to fill this void by observing and interviewing developers in order to understand their needs and better support their work.

Tool support for ML currently exists in two forms. *General* tools, such as Weka (Witten and Frank 2005), provide an extensive library of learning algorithm implementations. Although such tools make it very easy to conduct an experiment comparing the accuracy of different algorithms on benchmark datasets, they provide relatively poor support for the end-to-end development of a statistical ML component in the larger context of software development. *Accessible* tools, such as Crayons (Fails and Olsen 2003), focus on making it easy to develop a complete system. But this ease of use is attained by focusing on a particular domain and then hiding some of the complexity of the problem by limiting options available to a developer (such as limiting the developer to a pre-determined algorithm or limiting the developer to providing training data for a fully-automated system). A developer has little or no recourse if their problem does not conform to the assumptions made by the tool, as flexibility has been sacrificed for ease of use. This paper discusses general obstacles to accessibility and provides insight into creating new tools that are both *general* and *accessible*.

Studies

To understand the difficulties that developers encounter when integrating statistical ML into software, we first interviewed researchers experienced in applying statistical ML in applications. We interviewed statistical ML experts who have applied their expertise to HCI applications as well as HCI experts who have looked to use statistical ML to solve their hard problems. Based on results from our interviews, we then conducted a second study where participants built a simple application that used statistical ML. Over the course of five hours, participants worked on many aspects of a statistical ML system: they collected handwriting data, wrote Java code to extract features, and trained classifiers using Weka. We tested the performance of the resulting systems on data the participants did not have access to (2000 handwritten digits gathered from 20 people). A more detailed explanation of our studies can be found in our original publication (Patel et al. 2008).

Results

Based on our interviews and laboratory study, we distilled three related categories of difficulties: (1) difficulty following an iterative and exploratory process, (2) difficulty understanding the relationships between data and models, and (3) difficulty evaluating the performance of models in the context of an application. We summarize these difficulties here; please refer to our full paper for a more detailed analysis and discussion.

Following an Iterative and Exploratory Process

For many statistical ML problems, data flows through a system in a typical manner. Raw training data is collected and manipulated into a form from which features can be extracted. Features are then extracted, and statistical ML algorithms model the featurized data. During testing, raw test data is transformed into featurized data and the trained model is then applied. This flow of data is linear, as each step is dependent upon the output of previous steps.

In contrast to such a linear data flow, effective development is fundamentally iterative and exploratory. Unfortunately, developers inexperienced in applying statistical ML can confuse a linear data flow with a linear process. Furthermore, the boundaries between existing tools can create information gaps that impede exploration.

Developers can often spend an inordinate amount of time on one aspect of their problem, most commonly algorithm selection, while failing to identify errors and deficiencies in other aspects of their problem. One interview participant described a project where he and his collaborators spent too much time on algorithm selection: "We basically tried a whole bunch of Weka experimentation and different algorithms … nothing worked, so we decided that … maybe we should explore the feature space." Other interview participants found that they could move past an apparent dead-end by collecting additional data or by changing their problem definition. Based on these interviews, it is clear the effective application of statistical ML requires iteratively exploring all aspects of a system.

Iterative exploration is impeded by current tools that create information gaps when they encapsulate only a portion of the process of applying statistical ML. For example, Weka and other general tools are focused on modeling algorithm selection, but do not directly address feature development. They instead assume input in a general tabular format, with rows corresponding to instances and columns to features. This assumption makes it difficult or impossible to link the featurized data back to the raw data from which it was computed. There is thus no straightforward way to implement even simple debugging functionality, such as showing a developer the raw data associated with misclassified examples from an experiment. One of our interview participants, a statistical ML expert, lent some insight into the importance of debugging statistical ML systems. He reported using Matlab for much of his prototyping, stating "I think it's important to work in an interactive environment, [because] you can go back and ask a data structure 'what did you do?'" In contrast, tools that divide the process of applying statistical ML can prevent effective debugging of a system and can lead developers to become overly focused on the subset of the problem captured by the tool.

Another difficulty becomes evident over time. A developer may consider many different strategies for addressing a problem, but information about previous strategies is lost as a developer moves on. In order to track their progress, developers resort to brittle record-keeping. For example, one participant in our lab study resorted to naming a file "LogitBoostWith8to18EvenWindow-Iter=10.model." Four other participants kept handwritten paper notebooks tracking their progress. Such approaches have at least two

problems. First, it is often difficult to track everything that has changed between successive tests. Second, if a bug is found in feature generation or data collection, it becomes unclear whether previously explored strategies need to be revisited. Automatic tracking of exploration is currently difficult in part because tools address only portions of the process of applying statistical ML.

Understanding Data and Output

Our interviews show that developers new to using statistical ML tend to treat algorithms as "black boxes." They often expect algorithms to be immediately effective, and so they neither attempt to manipulate model parameters nor seek to understand how the algorithms work. The "black box" mindset prevents developers from understanding and debugging a system, and it is reinforced by opaque implementations of algorithms in current tools.

In the best case, a lack of understanding leads developers to randomly experiment with different algorithms, data, and features. In the worst case, a lack of understanding can mislead developers. One interview participant spent months training a classifier of forum posts, using an existing featurized dataset provided by another researcher. After getting the model to perform reasonably in cross-validation experiments, he found the model failed to generalize. Looking back at the raw data, he saw there had been a large spam attack on the forum, and all of the spam was of the same class in his training data. The classifier mistakenly learned about features correlated to spam, a mistake he may have detected earlier if his tools provided better insight into the basis for each classification.

In our laboratory study, the participants most experienced with statistical ML initially used simple algorithms to help gain an understanding of their features and data. In one case, a participant used decision trees to figure out which features were being selected and to understand the relationship between features and classifier output. Later in the session, after debugging his features, he shifted to using more complex and less interpretable algorithms. This staged experimentation process helped the participant make better informed decisions about his next step.

Current tools not only accommodate bad experimentation practice, but seem to encourage it. Because the primary feedback given by current tools is simple accuracy values, developers can mechanically manipulate a tool until it reports an acceptably high accuracy. Tools instead need to encourage developer understanding of statistical ML systems, so that developers can understand why their systems behave as they do and how they might take a principled approach to improving upon an existing system.

Evaluating Performance in Applications

Developers often find it difficult to evaluate the performance of statistical ML systems within the context of their application. While current tools provide extensive support for measuring accuracy, the success of an application is not captured solely by accuracy. Developers instead have other additional concerns. For example, one interview participant discussed privacy in his sensing application, stating "this was the kind of tradeoff between [more expressive features] and [privacy-sensitive features]." Further concerns include classification speed, training time, and how to incorporate the collection of training labels into an application. Often these factors need to be assessed within the context of the application itself, but current tools do not allow developers to easily export their entire data flow (i.e., from raw data to a model) for interactive testing within the context of an application.

Several interview participants also recalled difficulties they had experienced when developing statistical ML systems that needed to be robust to input from different people. For example, one participant complained "the cross-validation would show … 85% to 90% accuracy… and then [when] you would try it … it worked extremely well for some people and not well for others." Developers often assume that data gathered from a group of people will generalize to the entire population, and this may be true with a sufficiently representative group. However, large corpora are both hard to collect and expensive to process.

We explicitly examined this difficulty in our laboratory study by providing participants with data collected from four different people. Participants could have used this data to conduct experiments probing how well their systems performed when tested with people who had not provided training data (perhaps using an experiment that held out data from one person in each trial). Every participant instead used simple random cross-validation. One even explicitly commented that they probably should take advantage of having data from different people, but that they would instead use a random cross-validation because it was easier to do in Weka. We also note that participants significantly over-estimated the reliability of their models ($t(9) = 5.96$, $p < .001$). In fact, high performance led some of our participants to quit the task early when additional time may have led to better results. While random

	Accuracy Estimated by Participant	Accuracy on 2000 Test Digits	Error in Accuracy Estimate
P1	87.8%	84.7%	3.1%
P2	98.0%	74.3%	22.7%
P3	89.1%	78.3%	10.8%
P4	95.6%	82.9%	12.7%
P5	91.3%	84.7%	6.6%
P6	90.4%	78.0%	12.4%
P7	72.0%	56.9%	15.1%
P8	26.5%	22.8%	3.7%
P9	92.8%	78.8%	14.0%
P10	93.4%	84.4%	9.0%

Figure 1: Comparing estimates of model accuracy computed by participants to model performance on 2000 test digits. Participants significantly over-estimated model accuracy.

cross-validation is an important tool, tools encourage it even when it may not be appropriate. Tools instead need to help developers design better experiments.

Implications

As statistical ML continues to grow in importance, development tools need to enable and encourage effective practices. We now briefly present implications for building statistical ML tools, designing algorithms, and presenting statistical ML to developers.

Statistical ML Tools

The first step to building more effective tools is to remove the gaps within current tool chains. Tools might be integrated using lightweight metadata that preserves the origin of data as it is transformed. Another approach would be to create an integrated development environment that captures the entire process of building a statistical ML system. Tools need to support a developer's constant exploration of all aspects of a system, enabling the interactive debugging that is critical to the effective development of statistical ML systems. For example, a developer should be able to identify a misclassified instance in a confusion matrix and then trace back through every detail of the classification of that instance.

As integration becomes tighter, many new features become possible. A tool could log a developer's experimentation history: the data, features, and algorithms they explore. If an error is detected that invalidates a series of experiments (e.g., an error in feature computation code), experiments could be automatically rerun in the background. Automated systems could mine experimentation histories to suggest new experiments. A path through a history tree could represent the steps needed to go from raw data to a final model, and the code for that path could be automatically exported for use within an application.

Designing Algorithms

To effectively use statistical ML, developers need to understand how and why a system failed. They need to be able to diagnose and debug errors. However, many statistical ML algorithms are not designed to help developers understand classifiers. Algorithms are instead opaque and difficult to interpret. Even algorithms that lend themselves to human interpretability are presented by tools as "black-boxes" and treated as such by developers.

Prior work has focused on human-interpretable ML for end users, but the developer population has received relatively little attention. Developers attempting to use statistical ML are actively working with features and data in fundamentally different ways than end users. For example, finding relationships between features and erroneous behavior might lead a developer to identify and fix a bug in feature computation code. To promote the adoption of statistical ML by developers, greater emphasis needs to be placed on algorithm interpretability and interactivity.

Presenting Statistical ML

Interview participants noted that their goal is to "make a system work," but this is hard even when implementing an approach that has already been shown to work. Developers would benefit greatly from repositories containing information about what combinations of algorithms, features, and data work well together, as well as code they can then use in implementing their systems. Tool support could facilitate this by providing a common framework within which researchers could disseminate state of the art techniques to developers.

Developers would also benefit from additional help devising and following good experimental procedures. More could be done to clarify the strengths, weakness, and pitfalls of different algorithms. For example, a repository could present an algorithm, datasets on which it performs well, datasets on which it performs poorly, and a discussion of why it behaves the way it does.

Conclusion

As statistical ML continues to grow in importance, its methods and tools need to be made more accessible to software developers. This paper presents initial work in this direction by studying the development of statistical ML components of applications, by presenting three difficulties faced by developers, and by describing three solutions for making statistical ML more approachable.

Acknowledgements

We thank the many other people who contributed to this work. 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

Fails, J.A. and Olsen, D.R. (2003). Interactive Machine Learning. *International Conference on Intelligent User Interfaces (IUI 2003)*, 39-45.

Hand, D.J. (1998). Data Mining: Statistics and More? *The American Statistician, 52*(2), 112-118.

Ko, A.J., Myers, B.A., and Aung, H. (2004). Six Learning Barriers in End-User Programming Systems. *IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2004)*, 199-206.

Myers, B.A., Hudson, S.E., & Pausch, R.F. (2000). Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction (TOCHI), 7*(1), 3-28.

Patel, K., Fogarty, J., Landay, J.A., & Harrison, B. (2008). Investigating Statistical Machine Learning as a Tool for Software Development. *ACM Conference on Human Factors and Computing (CHI 2008)*, 29-38.

Witten, I.H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Francisco: Morgan Kaufmann.