Lowering the Barrier to Applying Machine Learning

Kayur Patel
Computer Science and Engineering
DUB Group
University of Washington
Seattle, WA 98105
kayur@cs.washington.edu

Abstract
Researchers have used machine learning algorithms to solve hard problems in a variety of domains, enabling exciting, new applications of computing. However, research results have not transferred to software solutions. In part, this is because developing software with machine learning algorithms is itself difficult. My dissertation work aims to understand why using machine learning is difficult and to create tools that lower the bar so that more developers can effectively use machine learning.

Keywords
Machine Learning, Software Development, Integrated Development Environments

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

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

General Terms
Human Factors

Introduction
Machine learning systems are used to address hard problems in a variety of domains. For example, researchers have used learning techniques to model human activity [3] and understand the human genome [5]. These tasks are high value: activity modeling allows us to track our environmental impact and genetic analyses allow us to better diagnose diseases. Moreover, these tasks are impossible without robust learning solutions. However, few applications take these research findings and turn them into software solutions. My research is motivated by the observation that much of this difficulty stems from the lack of appropriate tools -- few tools support the development of learning-based software. My goal is to develop new tools that enable the broad impact of machine learning by supporting developers as they iteratively design, test, and refine learning-based software. This is a critical challenge for advancing the role of machine learning in computer science research and practice.

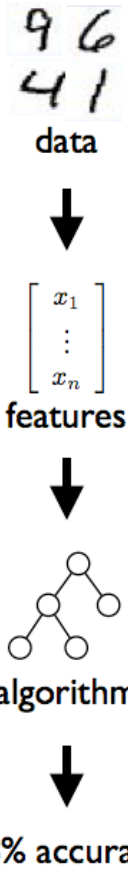

Figure 1: Applying machine learning requires collecting data, extracting features, training a model, and conducting experiments to evaluate a system.

My dissertation work aims to understand the deficiencies in current tools as well as propose and build new tools that can address the difficulties involved in implementing learning-based software. Specifically, my contribution is three-fold. First, I aim to understand the difficulties of using machine learning, not only in the choice of learning algorithms but also in the *entire process* from collecting data to testing a trained model. I have conducted interviews and laboratory studies observing how developers work with state-of-the-art tools, and I have distilled the process they use while building a machine learning system. Second, I aim to create new tools to address the deficiencies of current state-of-the-art tools. To this end, I have begun work on Gestalt, an integrated development environment (IDE) that supports the application of machine learning techniques. Third, I will show how the novel features of Gestalt help with the development of systems that leverage machine learning.

Understanding the Process

Figure 1 illustrates a classification pipeline that is typical of many machine learning applications. Data must be collected in some raw format, which is processed to extract feature vectors, which are used to train a model, which is then evaluated in experiments. To inform our design of new tools and understand the current process developers take, I conducted two studies that examined the challenges developers encounter when using existing tools [4]. First, I interviewed eleven researchers who had built learning-based software. The researchers described and diagrammed their processes, discussing not only successful strategies but also pitfalls and difficulties. I interviewed researchers with machine learning expertise as well as researchers relatively new to machine learning who were applying it in their research. I reasoned this mixture of expertise would uncover difficulties that people encounter in applying machine learning as well as best practices for overcoming those difficulties. My second study sought to further ground the results of my interviews through laboratory observations of actual work. Ten new participants each spent five hours building the machine learning component of a small application, a simple handwriting recognition engine. Participants provided input at all stages of the learning system: they collected data, wrote Java code to generate features, trained models using the Weka library, and conducted experiments to test the accuracy of their system.

From the results, I distilled three main difficulties that developers face when using machine learning. First, the studies show that the successful application of machine learning is generally based in an *iterative and exploratory* process. A developer examines all of the steps in the pipeline to find the step where they can make changes that will have the most impact on how well the entire system works. Second, the studies show that developers often have good intuitions about the individual links in their chain (e.g., their data and their features), but find it hard to *understand the relationship* between accuracy and these familiar parts of the machine learning system. Third, developers have difficulty *evaluating the learning system in the context of their application*. Developers are often concerned with more than just accuracy. For example, a developer of an embedded system may care about speed of feature computation and classification (because this code needs to run on the embedded device), and might be willing to make tradeoffs in classification accuracy related to this performance.

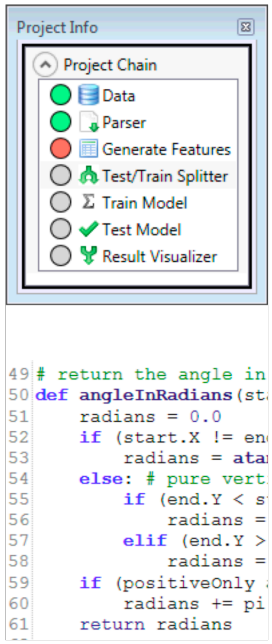

Figure 2: Gestalt provides flexibility through an explicit representation of the classification pipeline and flexibility by allowing developers to edit and change each step in pipeline by writing a python script.

Building Better Tools

Prior work within the HCI community has focused on building *structured* tools to help developers with a specific learning task. For example, the Crayons system, created by Fails and Olsen, makes it easy for developers and designers to create models for an image segmentation task [1], and Exemplar, created by Hartmann *et al.*, provides a direct manipulation approach for training systems based on sensors [2]. These tools are each based in an understanding of the problem they address, are built to support that problem, and typically attain their ease of use by limiting user contribution to a small set of points in the classification pipeline. In Crayons, for example, the only supported interaction is to color pixels. Structured tools are powerful approaches for solving their particular problems, but these tools lack flexibility to adapt to new or different scenarios.

A developer working on a new or different problem needs flexibility. General programming languages, such as Java, or integrated environments such as Matlab, are *flexible* enough to represent any problem. Support for these flexible systems consists of APIs, such as Weka, which provide standard working implementations of algorithms or features [6]. However, developers using these tools, especially developers without much machine learning experience, often do not know where to begin. Therefore, my goal is to create new tools that support the entire process of applying machine learning while providing the flexibility that developers need to solve their problems.

To address the deficiencies in current tools, I have begun work on Gestalt, a general environment supporting the application of learning techniques. Gestalt makes it easy to iterate on steps in the pipeline and to understand relationships between data, features, and results. Like most IDEs, Gestalt has standard text editing and project management capabilities. In addition, Gestalt provides *structure* through an explicit representation of the classification pipeline and *flexibility* by allowing developers to change individual parts of the pipeline by writing new scripts.

The first classification pipeline that developers build usually does not work. Our studies show that developers have to iterate on different steps in the pipeline to create a working solution. In structured tools, developers iterate by performing a few domain-specific actions. Flexible tools provide developers with more control but lack the feedback needed to determine what to do next. Connecting the data, features, and results can provide the feedback needed for developers to understand how the system is behaving so they can iterate. Current flexible tools make it hard to connect the steps in the classification pipeline. Gestalt provides explicit support for connecting the steps in the pipeline and provides interactive visualizations through which developers can quickly sort, filter, and color examples to drill down into the data they need.

The current version of Gestalt is focused on a single pipeline and a limited number of visualizations based on viewing raw data. In my dissertation work, I plan to add additional capabilities that focus on visualizations for other steps in the pipeline (e.g., feature generation) and provide support for more than one pipeline. For example, I plan to extend Gestalt to keep track of changes to the pipeline. This will allow developers to

compare different versions of a pipeline and tie their actions to changes in model performance.

Testing Gestalt

In my dissertation, I plan to study how Gestalt helps developers effectively use machine learning. I have already conducted a lab study looking at how Gestalt helps developers debug existing classification pipelines. Side-by-side visualizations of data, features, and results help developers understand learning bugs. In current tools, creating side-by-side visualizations is hard. Even a flexible environment like Matlab, which provides functionality for building visualizations, fails to provide support for connecting steps in the pipeline. In contrast, the visualizations in Gestalt are specifically designed to be connected.

In an initial test of Gestalt, I compared it to a baseline condition similar to Matlab. I created solutions for two different learning problems. Each solution was injected with bugs, and participants were asked to find the bugs using the features of the tool they were given. The study found that (1) all of our participants preferred Gestalt, and (2) they found more bugs and spent more time looking at visualizations when using Gestalt. Future studies will look at how Gestalt can provide better support for other difficult tasks, such as creating a new pipeline for an unknown problem or selecting the right set of features for a particular problem.

Contributions

In my dissertation, I will provide three specific contributions. First, I will contribute results from studies that describe the difficulties faced by developers using machine learning techniques. Second, I will create Gestalt, a development environment that lowers the bar for developing machine learning solutions. Finally, I will contribute results from studies evaluating the effectiveness of the methods presented in Gestalt.

Acknowledgements

Thanks to James Fogarty, James Landay, Steven Drucker, and Andrew Ko for guidance and advice.

Citations

[1] Fails, J.A. and Olsen, D.R. (2003). A Design Tool for Camera-Based Interaction. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2003), 449-456.

[2] Hartmann, B., Abdulla, L., Mittal, M. and Klemmer, S.R. (2007). Authoring Sensor-Based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2007), 145-154.

[3] Lester, J., Choudhury, T., Kern, N., Borriello, G. and Hannaford, B. (2005). A Hybrid Discriminative/Generative Approach for Modeling Human Activities. *International Joint Conference on Artificial Intelligence* (IJCAI 2005), 766-772.

[4] Patel, K., Fogarty, J., Landay, J.A. and Harrison, B. (2008). Investigating Statistical Machine Learning as a Tool for Software Development. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2008), 667-676.

[5] Wang, H., Segal, E., Ben-Hur, A., Koller, D. and Brutlag, D. (2004). Identifying Protein-Protein Interaction Sites on a Genome-Wide Scale. *Proceedings of the Conference on Neural Information Processing Systems* (NIPS 2004),

[6] Witten, I.H. and Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.