# Three research challenges at the intersection of machine learning, statistical induction, and systems

Moises Goldszmidt, Ira Cohen
*Hewlett-Packard Labs*
*Palo Alto, CA*

Armando Fox, Steve Zhang
*Computer Science Department*
*Stanford University*

## Abstract

Recent research activity [2, 12, 27, 10, 1] has shown encouraging results for performance debugging and failure diagnosis and detection in systems by using approaches based on automatically inducing models and deriving correlations from observed data. This paper explores research questions and preliminary results regarding the next steps in advancing this line of work. We specifically formulate three challenges. First, as new data is collected from a system, there is a need to continuously assess the validity of previously induced models, with the ultimate aim of achieving online adaption to system changes. The second challenge deals with interactions with human operators, including interpretation of model findings to generate explanations, enabling operator feedback to improve the models, and handling false positives and missed detections. The third challenge focuses on transforming the output of these models into structured representations or *signatures* of system state, so that they can serve as a machine-readable index for diagnosis and repairs. These challenges arise directly from the *application* of statistical and machine learning techniques to real-world systems, and we contend that through domain knowledge of this same application we may identify well-engineered solutions that allow the full potential of statistical and machine learning approaches to be realized.

## 1 Introduction

The complexity of today's deployed software systems is staggering, as is the rate of growth of that complexity. In terms of lines of code, in the last ten years Linux has grown by a factor of 30 and Cisco IOS by a factor of 10 while Apache has grown by a factor of five in the last five years. The result is that more than 90% of a typical corporate IT budget is devoted to administration and maintenance of existing systems [11] whose complexity surpasses human operators' ability to diagnose and respond to problems rapidly and correctly [17, 26].

Fortunately, promising initial results have been reported in using automatically-induced probabilistic and machine-learning-based models for problem localization [10], performance debugging [2, 1], capacity planning, system tuning [38], detecting non-failstop failures [27], and attributing performance problems to specific low-level metrics [12], among others. These efforts differ in the specific techniques, models, and assumptions (we list some representative examples later), but the general approach may be summarized as follows: Collect raw data from the running system; automatically induce a model over this data; use the model to make inferences. We believe this general direction is extremely encouraging because the automatic construction of models from data brings the promise of rapid adaptation to system changes or to unanticipated conditions. Despite the differences among approaches, we expect that there will arise fundamental challenges that *any* effort utilizing statistical methods will have to confront. Given the successes so far, we detail in this paper three such challenges in hopes of guiding this line of research towards realizing its full potential.

Our challenges may be summarized as: Can we design effective procedures and algorithms that continuously and automatically test the validity of models against a dynamic environment? How can model findings be interpreted by the human operators of the system, e.g. identifying false positives, converting model findings to actionable information, and possibly accepting feedback from experts? How can we maintain a long term, indexable, and searchable history of system issues, annotated in some cases with diagnosis/repair action, to leverage past diagnosis efforts and enable use of similarity-based search techniques in order to identify recurring problems and group similar problem incidents into common "syndromes"?

To understand how these challenges arose, it is use-

ful to review some concrete approaches, including their assumptions and methods (Section 2); we then explore each challenge in detail (Sections 3–5) and give an example of how the challenges is addressed in the context of a specific approach. We make concluding remarks in Section 6.

## 2 Early explorations

We highlight some recent specific successes of recent approaches that automatically induce models and correlate data from a running networked system. These approaches concentrate on transforming data into information that can be used to make decisions. Our intent is not to present a complete survey, but to outline the ways in which the different approaches map to real systems problems, the assumptions underlying this mapping, the consequences of violating those assumptions, and the similarities among the approaches that will motivate our fundamental challenges.

One set of approaches relies on modeling *normal* behavior and then identifying sufficiently large *deviations* as a possible indication of undesirable behavior. For example, the technique in [27] identifies rarely-occurring paths at runtime by using probabilistic grammars to model the distribution of *normal* paths of program execution at the software-module level. The assumption is that a sufficiently *anomalous* path may indicate a possible failure. When this assumption is violated, e.g. because a rare but legitimate code path is traversed, an automatic repair system might mistakenly take a repair action. The work discusses options for low-cost repair techniques that cause no harm if invoked by mistake, to mitigate such inevitable "false positives." In controlled experiments with realistic workloads, this technique detected 15% more failures than existing generic techniques; localization of these faults exhibits a classic recall/precision tradeoff, with false positive rates (1 − precision) approaching 20% for high values of recall, emphasizing the importance of dealing with false positives.

A second approach [12, 41] explicitly defines abnormal vs. normal behavior in terms of a directly measurable high-level objective, such as a threshold on response time or throughput and uses Bayesian network based classifiers [19] to capture the relationship between these objectives and low-level system metrics. When the high-level objectives are violated, the models determine which low-level metrics are correlated with the violation and which are not; this information can be used to identify likely causes of the performance problem. The assumption is that the Bayesian network classifiers do a good job of capturing patterns of low-level metrics that correlate well with violations of objectives; the approach provides a way of *scoring* its models so it can be determined when this assumption does not hold. Experiments with this approach, both on an experimental testbed and using data from a geographically distributed Enterprise production environment, showed that using a handful of inter-correlated metrics (between 3-8) is often enough to capture between 90-95% of the patterns of normal and abnormal behavior, and generally pointed towards a correct diagnosis/repair action of a performance problem.

A third approach, exemplified by [1], proposes algorithms to reconstruct the *causal* paths followed by transactions through the system, and then identifies path sites corresponding to high time consumption (i.e. possible performance bottlenecks). The assumption is that these causal paths can be reconstructed (in a statistical sense) using time precedence and regularities in the times between the different subtransactions. Note that in this case, there is no consideration of normal or abnormal system behavior. The intent is to provide visibility of the locations where time is spent in the different stages of the transactions. Preliminary results based on different types of traces provide evidence that the algorithms presented in [1] do produce useful and accurate results.

Finally, the work in [38] uses Influence Diagrams to model and tune the parameters for the Berkeley DB embedded database system. Results indicate that the proposed methodology is able to recommend optimal or near optimal tunings for a varied set of workloads including workloads that are not encountered during the model training phase.

Although the above approaches have shown promising initial results, they face some common challenges that are generally not addressed within the scope of the work so far. Even if the most general forms of some of these challenges remain open problems in machine learning, computational learning theory, or data mining, the obstacles may be surmountable for *specific applications* of these approaches to real systems problem with robust engineering solutions.

1. **Model validity:** How can we guarantee that at all times the model being applied is valid, i.e. that it usefully and correctly captures some essential characteristic of the system's operation? This is especially difficult when the behavior of the system being modeled changes dynamically and when both the training data and the "ground truth" for evaluating model accuracy are incomplete or noisy.
2. **Human in the loop:** How do the operators of such systems interpret what is reported by the model? This includes issues such as visualizing results, converting the model's findings to actionable information, dealing with false positives and false negatives, generating explanations, and enabling the insertion

of human expert knowledge and feedback into the models.

3. **Maintaining searchable history of models output** How can we represent the output of the models to enable search of past events and diagnosis/repair actions? This is important so that administrators can leverage past diagnosis efforts, identify quickly recurrent failure modes, among other needs.

## 3 Challenge 1: A valid model anytime

What should the metrics of "validity" be, given the challenges of determining the ground truth (required to evaluate the model) under the less-than-ideal conditions of a production environment? How do we know that the training data is sufficiently representative of the data seen during production operations—an implicit assumption of most of these approaches? Any realistic long-term resolution of these issues must provide a methodology as well as algorithms and procedures for managing the lifecycle of models, including testing and ensuring their applicability and updating their parameters.

This challenge is not inherent in machine learning itself; indeed, that literature is rife with methods for evaluating and estimating[1]the accuracy of models, and with metrics and scoring functions to compare different models against a dataset [16, 5, 22]. Moreover, statistics textbooks [32] provide algorithms for iterative loops comprising the steps and statistical tests for model evaluation, model diagnosis, and selection of remedial measures to repair the model (if possible). Model diagnosis involves checking whether the assumptions embedded in the models (e.g., linearity of the data, Gaussian noise) correspond to the data at hand; remedial measures may include enhancing the models with additional elements (e.g., metrics), or changing the type of model used (e.g., sets of linear regressions, or nonlinear elements).

Such procedures, while rigorous and well-defined, generally require human intervention to (sometimes visually) check the results of certain steps, adjust parameters and make decisions. The challenge is to automate this process as much as possible by taking advantage of our specific problem domain. A central aspect of this challenge in our domain stems from the complexity and dynamic behavior of the systems we deploy: changes in the system can occur frequently and at unpredictable times. Consequently, the machine learning procedures described above require online implementations so that models can be constantly updated to adapt to the changes in the system. Evidence of this need has been established in [12, 27] with different models and conditions.

[1]Since we cannot guarantee that all the pertinent data is available at training time, we can only produce an estimate of the accuracy of a classifier [28].

Various possible strategies to the model-validity problem might be considered:

1. Build an omniscient model capable of capturing all relevant behaviors. This goal may be unrealistic except in restrictive and benevolent environments. It assumes that at training time we would have access to enough data capturing *all* relevant behaviors.
2. Build a model with an identifiable set of parameters that can adapt to new data. Besides identifying the parameters themselves, this requires mechanisms for identifying the need for adaptation, executing the adaptation (i.e. adjusting the parameters), and data aging.
3. Rely on an ensemble of models. Different models in the ensemble are used in different situations. This requires mechanisms to select which model(s) to use in a given situation, decide when a new model must be added to the ensemble, merge inferences from different models, and discard obsolete models. One example of this approach is described in [41].

There is considerable work in machine learning, computational learning theory (COLT), and data mining addressing these issues (e.g. [9, 29, 4, 20]. The challenge is to adapt these approaches and enrich them with the particulars of our domain.

Another validity-related challenge involves estimating the amount of data required to build accurate models. Despite existing theoretical bounds and much recent progress [14, 25], results for representations such as Bayesian networks don't come easily [21] and researchers often resort to empirical estimation procedures. Although progress on this front has also been made in specific situations (e.g. [41]), we still lack well-engineered general approaches valid in the system domain.

Finally, validation of these models and techniques continues to be a major hurdle. In controlled settings, we may check some of the results by, e.g, injecting specific system conditions and verifying that they are correctly identified/diagnosed by the model. But in production systems, more often than not this "ground truth" will be unavailable, incomplete, or noisy. For example, an operator may suspect that some problem was being manifested in the system during some time period, but be unable to determine conclusively that a particular problem occurred at a particular point in time, or lack sufficient forensic data to reconstruct a problem and diagnose its true root cause (as was reported, e.g., in [7]). To make matters worse, more and more businesses may be willing to provide production data, but either unwilling or unable to provide the ground truth underlying that data, which is required to objectively measure the success of a method. In other communities, such as computer vi-

sion and bioinformatics, standard datasets have been collected and often manually analyzed, providing the means to objectively test and compare different machine learning methods. Such standard datasets are still missing in the system domain. We and our colleagues have called for the creation of an "open source"-like database of real annotated (but sanitized) datasets against which future research in this area could be tested [18], which could do for this line of applied research what the UC Irvine repository did to advance Machine Learning research [6].

## 4 Challenge 2: The human in the loop

Imagine a system administrator whose responsibility is to execute a triage as soon as system health or performance indicators indicate alarm. Depending on the outcome of the triage, the operator must call the system expert, application expert, network expert, or database expert. Ideally, the administrator would not only offer a justification for the triage and the decision to call a specific expert, but also provide possible explanations for the apparent system misbehavior. Such a scenario, which is quite common in real systems, illustrates that at various levels, humans with different knowledge and levels of expertise would be expected to interact with the models and their inferences. Can the models and their inferences be "interpreted" to generate the justifications and explanations that operators require?

In [12], the choice of Bayesian networks as the basic representation of a model was justified, in part, by the interpretability and modifiability of these models [23, 34]. It is also well known that decision trees can be used to generate "if-then" rules and part of the field of data mining concentrates on these issues [40, 35, 10]. These may provide initial building blocks, but much more research, engineering, and customizations are required to elevate these to the level of usable tools in the systems diagnostics domain.

We take as a given that the problems of false positives and missed detections (false negatives) will always exist. A major e-commerce site has reported false alarm rates in excess of 20% during normal operations. We therefore advocate research directed at minimizing their impact. A first step would be to translate the scores assigned to models during evaluation to a measure of confidence or uncertainty on the recommendations from these models. A second approach is to favor actions that are likely to have a salubrious effect if the alarm is genuine, but have relatively low cost if performed unnecessarily [8]. A framework for combining these ideas may be provided by casting the problem in decision theoretic terms: in this normative approach, the uncertainty of events, the cost/utility of repair actions, and the uncertainty of outcomes are combined to maximize expected utility (minimize expected cost) [34, 15].

In many cases, classifying an alarm as a false positive will still be the prerogative of the human operator. Can we design mechanisms and interfaces so that their expert knowledge can be used to enhance and improve the performance of these models, for example in helping them classify alarms rapidly? Can we also provide mechanisms so that feedback on model performance can be incorporated and used to change these models as appropriate? One strategy is to combine the formal models with other interpretive and diagnostic tools that play to the strengths of humans; for example, [7] presents evidence that combining anomaly detection with visualization allows human operators to exploit their ability for visual pattern recognition to rapidly classify an alarm as a false positive or genuine one. The challenge is to take the data generated by the many sensors, automatically filter out noise, find correlations, and display the information. Another method for using the human operator is known as *active learning*, a method in which the human is queried to provide additional information that would provide the most benefit in reducing false positives and missed detections [30, 13, 39].

## 5 Challenge 3: Querying the system's past

The third challenge we present concentrates on enabling the creation and management of a searchable history of the sytem's performance. The main benefits of this would be: (a) Similarity-based search for past diagnosis and repairs; (b) identification of recurrent problems[2]; and (c) groupings of problems to enable identification and prioritization.

We concur with Redstone et al. [37] that a first task is constructing a representation that captures the essentials of the system state for characterizing an undesirable (or desirable) observed behavior, and that can be generated in an automatic fashion. We will call this representation a *signature.* Signatures should be amenable to manipulation by computers, such as similarity based retrieval, and to annotation by experts with information regarding previous diagnosis and repairs; these abilities would enable the application of semi-supervised learning methods [31, 33] to improve the retrieval of proven solutions to recurring problems and identification of new problems. Signatures could also be subjected to automated clustering [16] to group similar problems into common "syndromes".

A primarily challenge, then, is to identify suitable similarity metrics to use in both retrieval and clustering. We attempted to generate signatures from the output of the

[2] Although we have concentrated on indexing undesirable system states, the same ideas can be used to capture "favorable" states.

probabilistic models described in [12, 41] for attributing application level performance problems to specific low-level metrics. During every 5-minute epoch, the models provide a list of system metrics that correlate with a violation of a performance objective, or a list of metrics whose values are abnormal in cases where the system is in compliance with the performance objective. These lists, plus additional information such as degree of correlation to the performance problem and other statistical related measures, are used as the signature. Though our prototype attempts to address the third challenge, in designing it we had to address the first two challenges as well.

Initially we used hand-labeled training data, and induced performance problems to confirm that our signature-generation method displays good similarity retrieval capabilities as well as good clustering properties. The "validity" challenge arose when we applied our technique to a production system. Decisions for the sizes of several windows of data had to be determined would have benefited from principled or well engineered methods for establishing the data needs for accurate models, and how these vary as the input varies. We were encouraged by the fact that we were able to use our signatures to identify all instances of a known problem. This problem took several weeks to identify as being recurrent, and generated over 170*KB* of text messages among geographically distributed system operators. Our signatures identified other sets of multiple incidents as potentially belonging to a single "syndrome" (recurring problem), but since the data corresponding to observed performance problems was only partially labeled, we continue to work with the operators to attempt to determine whether these findings and groupings are actually correct.

The "human in the loop" challenge was evident in our struggle to find visualization mechanisms to help operators compare different signatures. In addition, we still lack a systematic way to incorporate operators' expertise back into our methods. These problems are further confounded by the fact that responsibility for different tiers of our production system (application server, database, etc.) spans organizational boundaries across which there are differing techniques for data collection, troubleshooting, and alarm handling.

## 6 Conclusions

Recent research has demonstrated that it is possible to automatically induce models from raw data collected from a running networked system, and that these models can indeed transform raw data into useful information for many tasks related to performance debugging and isolation, anomaly detection, detecting and localizing non-failstop failures, among others. We are excited by the potential of these approaches in increasing the efficacy and efficiency of the management of complex IT systems.

With IT budgets dominated by human operator costs, the potential benefits would be significant even if these techniques only served to increase the effectiveness of less-experienced operators. We believe, however, that even experts will benefit from being able to quantify their intuitions about correlations, breaking points, and patterns of behavior. In addition, the possibilities of exploring the data efficiently will provide tools for testing new hypotheses and "what-if" scenarios.

We do not, of course, advocate statistical, probabilistic modeling, and pattern recognition techniques as the solution to all "self-*" problems. Beyond the well-known limitations of the benefit of automation and the problem of "automation irony" [36], the essence of the proposed research agenda is to understand the particular limitations of statistical approaches as applied to system problem detection, localization, and ultimately diagnosis. In order to understand these limits, we must identify the fundamental challenges that will be faced by any work in this area. We have attempted to formulate three such challenges and show how they arise in real problem instances. With the availability of high-quality open-source implementations of statistical induction and pattern recognition algorithms [3, 24, 40] and increasing interest in the integration of measurement frameworks with system middleware, now is the time to vigorously pursue this line of research and identify the limits and benefits of these approaches.

## 7 Acknowledgments

We thank P. Bodík, J. Chase, T. Kelly, E. Kıcıman, J. Mogul, D. Patterson, J. Symons, and M. Verber for discussions that contributed to these ideas, and the anonymous referees whose comments greatly improved the paper.

## References

[1] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In *19th Symposium on Operating Systems Principles*, 2003.

[2] P. Barham, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modeling. In *6th Symp. on Operating Systems Design and Implementation* , 2004.

[3] Bayesian network classifier toolbox. `http://jbnc.sourceforge.net/`.

[4] J. Binder, D. Koller, S. J. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29, 1997.

[5] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[6] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[7] P. Bodík, G. Friedman, L. Biewald, H. Levine, G. Candea, A. Fox, M. I. Jordan, D. Patterson, K. Patel, G. Tolle, and J. Hui. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In *International Conference on Autonomic Computing*, 2005.

[8] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. A microrebootable system – design, implementation, and evaluation. In *6th Symposium on Operating Systems Design and Implementation*, 2004.

[9] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, and M. Warmuth. On-line prediction and conversion strategies. In *Computational Learning Theory: Eurocolt*, 1993.

[10] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing*, 2004.

[11] T. Chou. *The End of Software*. Sams Publishing, Indianapolis, IN, 2005.

[12] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. Chase. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *6th Symp. on Operating Systems Design and Implementation*, 2004.

[13] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, 7, 1995.

[14] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[15] M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[16] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[17] A. Fox and D. Patterson. Self-repairing computers. *Scientific American*, June 2003.

[18] A. Fox, D. A. Patterson, and M. I. Jordan. Reliable adaptive distributed systems (research proposal). National Science Foundation Award, June 2005.

[19] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29, 1997.

[20] N. Friedman and M. Goldszmidt. Sequential update of Bayesian network structure. In *International Conference on Uncertainty in Artificial Intelligence*, 1997.

[21] N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian network. In *International Conference on Uncertainty in Artificial Intelligence*, 1996.

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2001.

[23] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20, 1995.

[24] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3), 1996.

[25] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT press, 1994.

[26] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[27] E. Kıcıman and A. Fox. Detecting application-level failures in component-based internet services. Submitted, Sept. 2004.

[28] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, 1995.

[29] T. Lane and C. E. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *International Conference on Knowledge Discovery and Data Mining*, 1998.

[30] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4), 1992.

[31] T. Mitchell. The role of unlabeled data in supervised learning. In *Proc. of International Colloquium on Cognitive Science*, 1999.

[32] J. Neter, M. Kutner, C. Nachtshein, and W. Wasserman. *Applied Linear Statistical Models*. McGraw-Hill, 1996.

[33] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39, 2000.

[34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[35] J. Quinlan. *C4.5 Programs for machine learning*. Morgan Kaufmann, 1993.

[36] J. Reason. *Managing the Risks of Organizational Accidents*. Ashgate Publishing Co., 1997.

[37] J. Redstone, M. Swift, and B. Bershad. Using computers to diagnose computer problems. In *Proc. 9th Workshop on Hot Topics in Operating Systems*, 2003.

[38] D. Sullivan. *Using probabilistic reasoning to automate software tuning*. PhD thesis, Harvard University, 2003.

[39] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *17th International Conference on Machine Learning*, 2000.

[40] I. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.

[41] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *Intl. Conf. on Dependable Systems and Networks*, 2005.