Integrating Symbolic and Statistical Methods for Testing Intelligent Systems

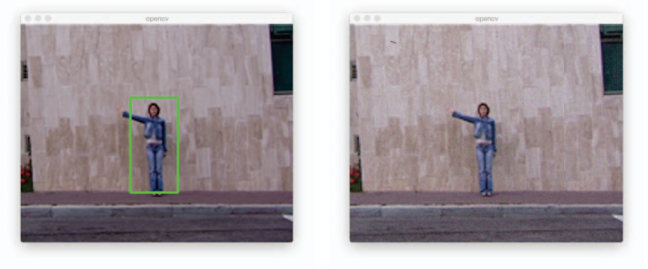
Applications to Machine Learning and Computer Vision

Arvind Ramanathan and Laura L. Pullum
Computational Science and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
Email: {ramanathana, pullumll}@ornl.gov

Faraz Hussain, Dwaipayan Chakrabarty, and Sumit Kumar Jha
Computer Science Department
University of Central Florida
Orlando, Florida, USA
Email: {fhussain, dchakra, jha}@cs.ucf.edu

***Abstract*—Embedded intelligent systems ranging from tiny implantable biomedical devices to large swarms of autonomous unmanned aerial systems are becoming pervasive in our daily lives. While we depend on the flawless functioning of such intelligent systems, and often take their behavioral correctness and safety for granted, it is notoriously difficult to generate test cases that expose subtle errors in the implementations of machine learning algorithms. Hence, the validation of intelligent systems is usually achieved by studying their behavior on representative data sets, using methods such as cross-validation and bootstrapping.**

In this paper, we present a new testing methodology for studying the correctness of intelligent systems. Our approach uses symbolic decision procedures coupled with statistical hypothesis testing to validate machine learning algorithms. We show how we have employed our technique to successfully identify subtle bugs (such as bit flips) in implementations of the k-means algorithm. Such errors are not readily detected by standard validation methods such as randomized testing. We also use our algorithm to analyze the robustness of a human detection algorithm built using the OpenCV open-source computer vision library. We show that the human detection implementation can fail to detect humans in perturbed video frames even when the perturbations are so small that the corresponding frames look identical to the naked eye.



(a) (b)

Fig. 1. (a) A human detection algorithm based on the histogram of oriented gradients (HOG) implemented using the OpenCV library correctly reports that the video frame on the left has a human. (b) The same OpenCV implementation of the human detection algorithm reports that the video frame on the right has no human. The video frame on the right is obtained by adding tiny perturbations to the left video frame. Both frames look identical to the naked eye.

I. Introduction

The formal verification of computer programs is a challenging problem due to the undecidability of even simple properties, such as program termination [1]. Despite such theoretical barriers, great strides have been made in the practice of software verification and validation [2]. Reachability and termination analysis of intricate software, including device drivers and operating system kernels, have made computer systems more reliable and suitable for high-assurance applications [3]. This practical success has been achieved through innovations in automated abstraction techniques as well as an exponential improvement in the capabilities of lightweight theorem provers and satisfiability solvers over the last few decades [4], [5].

Probabilistic programs [6], such as those implementing machine learning algorithms, pose a challenge to existing verification and validation algorithms [7] because of two reasons:

- Such programs are generally nondeterministic, i.e. they demonstrate different behaviors on the same input [8]. It is acceptable for machine learning systems to sometimes produce incorrect results, as long as their behavior can be considered correct for a significant part of the possible input space. Verification algorithms that attempt to disprove a program's correctness by automatically identifying incorrect behaviors on certain input cases (i.e. by searching for counterexamples) are therefore not useful for the analysis of machine learning algorithms. Hence, algorithms for validating intelligent systems must expose those regions of the input space that produce an *unacceptably-flawed output distribution*.
- Machine learning approaches often involve close integration of nonlinear data manipulation and control flow. This tight coupling makes it difficult to replicate the success obtained in the validation of other software and hardware systems such as device drivers and hardware controllers, the key to which is often the creation of smaller, discrete abstractions of the system to make it more amenable to algorithmic analysis [9].

In this paper, we study the generation of test cases that

978-3-9815370-7-9/DATE16/

can be used to verify the robustness of intelligent systems. Our approach uses a combination of symbolic decision procedures and statistical hypothesis testing [10] to *identify test cases where the intelligent algorithm produces an incorrect answer with an undesirably high probability*. Our method also introduces a distance-theoretic abstraction methodology that untangles the relationship between nonlinear data manipulation and control in intelligent systems. We apply our methodology to two prototypical machine learning problems – k-means clustering and human detection in computer vision. We demonstrate the success of our approach by detecting subtle errors, such as bit-flips, in an implementation of the k-means algorithm that are not readily detected using traditional approaches such as randomized testing. We also analyzed an implementation of the human detection algorithm built using the OpenCV open-source computer vision library [11], and showed that it failed to detect humans when provided video frames with only a small perturbation (see Figure 1). We observe that both images in the figure look almost identical to the naked eye, but the machine learning human-detection algorithm worked correctly on Figure 1a but failed on Figure 1b.

II. Related Work and Current Challenges

A. Testing Machine Learning Algorithms

Machine learning researchers and practitioners usually rely on intelligently designed unit and integration tests that provide test cases for a large part of the input space [12]. Due to the intrinsic probabilistic nature of machine learning algorithms, much of the testing framework is based on statistical analyses [13]. Our ongoing work [14] explicitly examines algorithms using the following verification and validation approaches:

- Cross-validation is used to estimate how the results can generalize to an independent dataset [15],
- Information theoretic measures that distinguish two algorithm implementation approaches are used to identify accurate (or more stable) version(s) of the algorithm,
- Metamorphic testing techniques are used to to find logical bugs [16], and,
- Fault injection methods are applied to provide statistical measures of robustness [17].

The development of automated techniques for the formal verification of hardware and software systems has been an important success story in the area of formal methods [18]. The main idea is to model the system of interest in a formal model description language, express the desired property to be checked in suitable mathematical logic, and then deploy algorithmic procedures to explore if the system satisfies the property of interest [19].

The verification procedure exhaustively searches the entire state space of the system explicitly or implicitly in order to validate if the property will always be satisfied by the system. If the property does not always hold in the system, the verification procedure usually provides a counterexample – the sequence of inputs and variable values that cause the property to be violated [20]. Since a large fraction of both natural and man-made systems of interest are stochastic, a variety of algorithms and tools for the verification of *probabilistic* systems have been developed [21]. In order to overcome the state-space explosion problem of large systems [22], modern formal verification tools often use *statistical* methods to solve the probabilistic verification problem [23]. We note that no universally-adopted framework for *formally verifying* machine learning systems has been presented so far.

B. Challenges for big data analytics and machine learning

Most data analytics and machine learning algorithms are *implemented to work with specific datasets* that obviously have certain characteristics unique to them. Therefore, the results observed on one dataset may not necessarily be translate with the same level of guarantee across other datasets. Although many scientific domains such as image processing, computer vision, and natural language processing, provide several standard benchmarks for testing specific learning algorithms, the diverse nature of datasets and the probabilistic algorithms used in the analytics and machine learning tools make it exceedingly challenging to objectively and quantitatively evaluate them [24].

An associated challenge with analytics algorithms is the inability to provide guarantees on their performance, for an *arbitrary* dataset. In particular, it is generally difficult to quantitatively reason why a specific algorithm provides a certain performance, either in terms of its accuracy, precision, recall and sensitivity [25]. For high-assurance and safety-critical applications it is necessary to obtain insights into the following questions:

- What are the conditions under which an algorithm will fail to provide correct answers?
- What are the conditions under which two algorithms developed to meet the same specifications, will provide different answers, given the same dataset?

The purpose of the above discussion is to (i) highlight why machine learning techniques need to be formally verified and validated, (ii) draw attention to the special challenges faced when dealing with the testing and validation of data analytics algorithms, and (iii) motivate why newer verification and validation techniques need to be developed specifically in the context of machine learning algorithms.

III. Symbolic Decision Procedures and Statistical Hypothesis Testing

Our validation approach uses a combination of symbolic decision procedures and statistical hypothesis testing to search for test cases where the behaviors of an intelligent system and its erratic variant can be shown to be statistically different.

Symbolic analysis of intelligent systems is hard primarily due two different sources of complexity: (i) the number of data points, and (ii) the *computationally expensive calculation of distances* between pairs of data points. It is widely known that a large number of "nonlinear" operations like calculating inter-point distances slow down symbolic decision procedures, such as bit-vector SMT solvers and Boolean Decision Diagram packages.

A. Main ideas behind our algorithmic approach for testing

The central idea behind our algorithmic technique for testing intelligent systems is to use a *combination of symbolic and statistical approaches* to *efficiently find test cases where minor changes in the input cause the intelligent algorithm to fail* by reporting a widely different answer. We rely on two important ideas to search for such test cases:

- We use modern automated *symbolic decision procedures* [5] to exhaustively search for test cases for which the intelligent system gives the correct answer in general, but is sensitive to even minor changes in the inputs.
- Since intelligent systems invariably are stochastic (i.e., they are allowed to make mistakes), we rely on statistical hypothesis testing [10], [26] to to determine, up to a desired degree of confidence, whether we have arrived at a test case that can reliably be used to distinguish between a correct and an erroneous intelligent system.

A correct system intelligent system would be robust, i.e. not excessively sensitive to minor changes in the input. An erratic intelligent system, especially in the case of machine learning algorithms, is likely to have been designed using limited or unrepresentative training data, and hence would be more prone to giving the wrong answer even when there are very minor perturbations in the input.

The next two subsections develop the formal background about our decision-procedure based search technique for generating test cases and present results on statistical hypothesis testing. We also describe a theoretical framework for defining abstractions of distances between points that help identify sets of points essentially equivalent to one another modulo translations and rotations. In Section IV, we discuss experimental evidence that a number of subtle errors like bit flips can be detected by the construction of test cases that consist of only a small number of data points.

B. Metric Abstractions

In this section, we present a theoretical framework for building distance-theoretic abstractions that obviates the need for the expensive computation of distances by the symbolic decision procedure. Introducing these efficient abstractions facilitates the task of finding test cases that can distinguish between a correct intelligent system and an erroneous intelligent system.

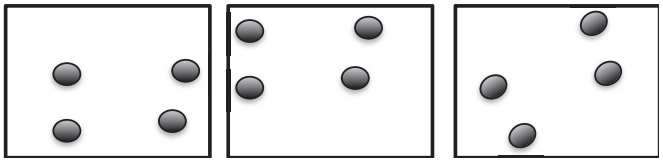


Fig. 2. Sampling points in the co-ordinate space can produce different sets of points that are *translationally or rotationally equivalent*. Thus, they drive many clustering and other learning algorithms along the same control path. We are using symbolic decision procedures to efficiently search for a set of data points that can distinguish between a correct and an erroneous intelligent system. Therefore, we need to avoid exploring paths (i.e. sets of input points) that are equivalent in their behavior with respect to the intelligent system. Our distance theoretic abstraction (see Definition 1) describes sets of points using a canonical representation that avoids the repeated analysis of equivalent paths, subject to translations (e.g., the leftmost two images shown above) or rotations (e.g., the leftmost and the rightmost images shown above).

A labeled set of points is a pair (Q,L), where Q is a finite set $\{q_i \mid q_i \in \mathbb{R}^m, 1 \leq i \leq N\}$ of N points in m-dimensional space and L is a labeling function that assigns an integer label $L(q) \in \{1,2,\ldots,M\}$ (where $M \leq N$) to each point $q \in \mathbb{R}^m$; usually $M \ll N$.

First, we present a new abstraction for verifying intelligent systems that relies on distance theoretic representation of a labeled set of points. In particular, if a set of points can be produced from another set of points using rigid-body motion (translation or rotation), the two sets of points have the same *canonical* representation in our abstraction. Further, our abstraction explicitly records the pairwise distance between the labeled points, and obviates the need for computing this distance while performing a symbolic analysis of the intelligent system.

Definition 1 (Metric Abstraction)**.** *The metric abstraction of a set of labeled data points (Q,L) is the triple (X,D,L'), where*

1) *$X = \{1,2,\ldots,n\}$ denotes the corresponding points in $Q = \{q_1,q_2,\ldots,q_n\}$,*
2) *D is the set of pairwise Euclidean distances between all pairs points in Q, i.e. $D = \{d_{ij} : d_{ij} = ||q_i - q_j||_2,\ i,j \in X\}$,*
3) *L' is the labeling function that labels elements in X with the label of the corresponding point in Q, i.e. $L'(i) = L(q_i)$.*

The existence of a metric abstraction for every set of points in Euclidean space is obtained directly from the fact that Euclidean space is a metric space. Next, in Lemma 1, we show that every metric abstraction corresponds to a set of points in Euclidean space such that the pairwise distance between these points is approximately the same as the distances described in the metric abstraction.

Lemma 1 (Approximate preservation of points by metric Abstraction)**.** *Given a metric abstraction (X,D,L'), there exists a cluster (Q,L), where*

1) *$Q = \{q_i \mid q_i \in \mathbb{R}^m, 1 \leq i \leq N\}$, such that the pairwise distances between all points is approximately preserved i.e. $(1-\varepsilon)d_{ij} \leq \sqrt{||q_i - q_j||_2} \leq (1+\varepsilon)d_{ij}$ for some small ε,*
2) *$L(q_i) = L'(i)$,*

Proof. First, the definition of the metric abstraction ensures that the distance measure D is a metric. Second, given a set of points X and a distance measure D between them that satisfies the following conditions:

1) $d(x_i,x_j) = 0$ if and only if $i = j$,
2) $d(x_i,x_j) = d(x_j,x_i)$, and
3) $d(x_i,x_j) + d(x_j,x_k) \geq d(x_i,x_k)$,

there exists an embedding E [27] into the space $\mathbb{R}^m$, where $m = O(\frac{\log n}{\varepsilon^2})$, such that $(1-\varepsilon)d(x_i,x_j) \leq \sqrt{||E(x_i) - E(x_j)||_2} \leq (1+\varepsilon)d(x_i,x_j)$. Choosing q_i as $E(x_i)$ completes the proof. □

C. Statistical Hypothesis Testing

Given an intelligent system $\mathcal{A}$ and a model describing a possible error in the implementation of the intelligent system $\mathcal{A}_e$, a natural idea would be to search for an input set of data points (Q,L) such that the erratic system always produces a different output as compared to the correct system:

$$\exists Q,L \text{ such that } \mathcal{A}(Q,L) \neq \mathcal{A}_e(Q,L)$$

For ease of presentation, we have assumed that the intelligent system produces a single scalar output. However, as the intelligent system $\mathcal{A}$ is probabilistic and the error being investigated likely occur only in rare instances, such a query will often be unsatisfiable and not produce any useful test cases. Hence, we suggest the use of a statistical test [28] to ensure that the *sample mean* of the correct intelligent system $\mathcal{A}$ is different from the

Inputs : Intelligent System $\mathcal{A}$
Erratic Intelligent System $\mathcal{A}_e$
Significance Level $\rho \in (0,1)$
Approximation Parameter $\varepsilon \in (0,1)$
Maximum number of iterations *MaxIter*

Outputs: Labeled Input Points (Q,L), where $Q = \{q_1,\ldots,q_N\}$ and $L : Q \to \{1,2,\ldots M\}$ such that $\mu_{\mathcal{A}(Q,L)} \neq \mu_{\mathcal{A}_e(Q,L)}$

```
/* Initialize input points and labels */
ExploredDistances ← ϕ /* Initialize distance */
ExploredLabels ← ϕ /* Initialize labels */
i ← 0 /* Initialize counter */
p ← 0 /* Initialize p-value */
while (p < ρ) /* Loop while p-value is less than
    significance level */
do
    i ← i+1
    /* Choose new distances and new labels */
    Choose pairwise distances D_i = {d_i1, d_i2, ... d_iN}
    between N points and the labeling function L_i such
    that D_i ∉ ExploredDistances or L_i ∉ ExploredLabels
    /* Update distances and labels */
    ExploredDistances ← ExploredDistances ∪ {D_i}
    ExploredLabels ← ExploredLabels ∪ {L_i}
    /* Compute new points from new distances using
       symbolic decision procedures or metric
       embeddings */
    Compute Q_i = {q_i1, q_i2, ... q_iN} such that
       (1−ε)||q_ij − q_ik||^2 ≤ d^2_ijk ≤ (1+ε)||q_ij − q_ik||^2
       /* Perform sequential sampling using this
       set of labeled points */
    n ← 0 /* Initialize number of i.i.d. test
       samples */
    while (p < ρ && n < MaxIter) /* Loop while
       p-value < significance level or maximum
       number of iterations is exceeded */
    do
        n ← n+1
        B[n] ← Random() /* New random source */
        /* Test both implementations using the new
           random source */
        x1[n] ← A(Q_i, L_i, B[n])
        x2[n] ← A_e(Q_i, L_i, B[n])
        /* statistical hypothesis testing */
        x̄1 ← (Σ_{l=1}^{n} x1[l]) / n
        x̄2 ← (Σ_{l=1}^{n} x2[l]) / n
        σ1 ← Standard-Deviation(x1[1], ..., x1[n])
        σ2 ← Standard-Deviation(x2[1], ..., x2[n])
        t ← (x̄1 − x̄2) / sqrt((σ1^2 + σ2^2)/n)
        p ← Compute p-value(t, n)
    end
end
Return (Q_i, L_i) and n
```

Algorithm 1: Testing Intelligent Systems using Symbolic Search and Statistical Hypothesis Testing

incorrect intelligent system $\mathcal{A}_e(Q,L)$ under different sources of randomness B_i:

$$\exists Q,L \text{ such that } \frac{\sum_{i=1}^{Z} \mathcal{A}(Q,L,B_i)}{Z} \neq \frac{\sum_{i=1}^{Z} \mathcal{A}_e(Q,L,B_i)}{Z}$$

Here, we use the notation $\mathcal{A}(Q,L,B)$ to explicitly capture the dependence of the intelligent system $\mathcal{A}$ on the source of randomness B. Let μ be the true mean output produced by the intelligent system $\mathcal{A}$ and μ_e be the true mean output produced by the intelligent system $\mathcal{A}_e$ under the error model e. Based on a sample of the executions of the intelligent system, we need to reject one of the following two hypotheses:

$$\text{Null Hypothesis: } \mu_{\mathcal{A}} = \mu_{\mathcal{A}_e}$$

$$\text{Alternate Hypothesis: } \mu_{\mathcal{A}} \neq \mu_{\mathcal{A}_e}$$

In order to conduct this hypothesis test [26], we compute the following t-score statistic [29]:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{(\sigma_1^2 + \sigma_2^2)/n}}$$

Here, $\overline{x_1}$ and $\overline{x_2}$ are the sample means for the intelligent systems $\mathcal{A}$ and $\mathcal{A}_e$, σ_1 and σ_2 are the sample standard deviations for for $\mathcal{A}$ and $\mathcal{A}_e$, and n is the sample size for both the samples. Then, we compute the p-value that the algorithm and its variant will produce different means under the input (Q,L). If the p-value is greater than a level of significance (say, 99%), we stop and report this input (Q,L) as a disambiguating test case that can distinguish the algorithm $\mathcal{A}$ from its erratic variant $\mathcal{A}_e$.

In summary, our primary contribution is a *new algorithmic technique for testing intelligent systems* that uses symbolic decision procedures and statistical hypothesis testing for *automatically generating test cases* that distinguish between a correct intelligent algorithm and an erroneous intelligent system.

IV. Case Studies

We have studied the validation of two prototypical intelligent systems: (i) the k-means clustering algorithm [30], and (ii) the Histogram of Oriented Gradients (HOG) human detection algorithm [31].

A. K-means Clustering

We tested the efficacy of our algorithm by trying to create test cases that can expose subtle errors such as bit-flips in the k-means clustering procedure [30]. The results shown in Table I demonstrate that randomized testing was not able to generate test-cases that could expose bit-flips after observing 100 samples. Using our symbolic analysis approach coupled with statistical hypothesis testing, we were able to distinguish between bit-flips using fewer than 100 samples.

In order to test the effectiveness of our algorithm, we sought to obtain test cases that allowed us to detect bit-flips in inputs. The results of our experiments are presented in Table II. Our symbolic testing algorithm was only allowed to generate at most 100 samples using the metric abstraction introduced in Section III-B.

In every case, we obtained a test case that differentiated the correct implementation from the one where a bit of one of the co-ordinates of one of the points was flipped with at least 70% probability i.e. a majority of the results in one case suggested one clustering while a majority of the results in the other

TABLE I

RESULTS OF RANDOMIZED TESTING FOR BIT-FLIPS ON THE K-MEANS ALGORITHM. THE ATTEMPT WAS TO FIND A TEST CASE WHERE FLIPPING A SINGLE BIT OF ONE OF THE INPUT POINTS WOULD CAUSE THE K-MEANS ALGORITHM TO FAIL. THE TABLE BELOW SHOWS THAT NO SUCH SET OF RANDOMLY SELECTED INPUT POINTS COULD BE FOUND EVEN AFTER REPEATED TRIALS.

Error Model			Random testing	
Point	Co-ordinate	Bit flipped	#Test Cases	Found?
1	X	1	100	NO
1	X	2	100	NO
1	X	3	100	NO
1	X	4	100	NO
1	Y	1	100	NO
1	Y	2	100	NO
1	Y	3	100	NO
1	Y	4	100	NO

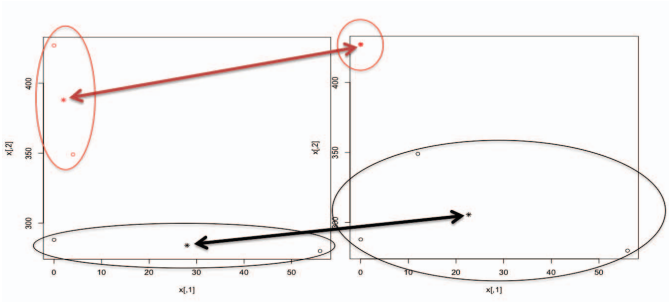


Fig. 3. **The impact of a single bit-flip on the clusters produced by the k-means algorithm.** A single bit-flip in the x-coordinate of only one data point changes the clustering produced. The left subfigure shows the original inputs and results of applying k-means; the right subfigure depicts the results of applying k-means when a single bit of one input point was flipped. Red and black ellipses indicate the clusters detected in each case, and the asterisks denote the cluster centroids. The colored arrows indicate how the cluster centroid have shifted.

case suggested a different clustering. An illustrative example is shown in Figure 3. The figure shows that a bit-flip in the input caused a drastic change in the clustering produced by the k-means algorithm.

B. Human Detection in Computer Vision

We also applied a combination of symbolic decision procedures and statistical model checking algorithm to the problem of detecting humans in video streams using the open-source implementation of the Histogram of Oriented Gradients (HOG) in the OpenCV computer vision library [31]. HOG was invented for the purpose of detecting pedestrians in static images, and has since been extended to include human detection in videos. It is a robust human detection descriptor that reported essentially zero miss rate on the MIT human database and a 0.1 miss rate on the INRIA dataset. The descriptor works by dividing each image into small connected regions and computing the histograms of gradient directions in each of these regions. Since the descriptor normalizes intensity across a large region of the image, it can adapt to changes in illumination and shadowing.

As the HOG detector is optimized for variations in lighting, simply adding a large amount of noise to the frame does not force an error. Figure 5 shows two examples of extreme illumination – very bright and very dark video frames – where the OpenCV HOG human detection works perfectly well.

(a) (b)

Fig. 4. Two video frames generated by our algorithm where OpenCV fails by reporting that no human is present in the frame. In both these cases, a human viewer can easily ascertain the presence of a human being in the frame.

(a) (b)

Fig. 5. Two extreme examples of video frames where OpenCV *correctly* reports that a human is present even though the frame has been changed substantially. This demonstrates that stress-testing the robustness of such computer vision systems is not just a matter of introducing a high-magnitude perturbation into the frame.

Figure 4 illustrates the perturbed video frames generated by our algorithm such that the OpenCV's implementation of the HOG human detection implementation fails to detect humans in these frames even when the perturbations are so small that the corresponding frames look identical to the human eye.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have made three contributions. *First*, we introduced a new way of using symbolic analysis and statistical testing to identify test cases that can expose subtle bugs in intelligent systems. *Second*, we showed how distance-theoretic abstractions can be used to identify test cases that can expose bit-flips in implementations of the k-means algorithm. *Third*, we demonstrated that subtle changes in video frames that cannot be detected by the human mind can cause the OpenCV human detection algorithm to produce incorrect results.

As machine intelligence is being introduced into safety-critical systems, systematic testing of machine learning systems is a topic that demands greater attention from the formal verification community. Our future efforts will focus on understanding the interplay between the nonlinear data computations and the control flow in different learning algorithms so that efficient abstractions can be tailored for individual algorithms. Our

TABLE II

RESULTS OF SYMBOLIC TESTING FOR BIT-FLIPS ON THE K-MEANS ALGORITHM. THIS TABLE SHOWS THE INPUTS FOR WHICH A SINGLE BIT-FLIP IN ONLY ONE OF THE DATA POINTS CAUSES K-MEANS TO FAIL BY REPORTING A DIFFERENT CLUSTERING. COMPARE THE RESULTS HERE WITH TABLE I THAT DOCUMENTS THE UNSUCCESSFUL ATTEMPTS AT RANDOMIZED GENERATION OF SUCH TEST CASES.

Error Model			Test	Clustering Result	
Point	Co-ordinate	Bit flipped	Input	Correct	Incorrect
1	X	1	{(134, 0), (64, 0), (100, 30), (0, 0)}	(2 2 2 1)	(2 1 2 1)
1	X	2	{(133, 0), (64, 0), (100, 30), (0, 0)}	(2 2 2 1)	(1 2 1 2)
1	X	3	{(131, 0), (64, 0), (100, 30), (0, 0)}	(2 2 2 1)	(2 1 2 1)
1	X	4	{(549, 1792), (564, 1713), (529, 1678), (640, 1858)}	(2 1 1 2)	(2 2 2 1)
1	Y	1	{(549, 1793), (564, 1713), (529, 1678), (640, 1858)}	(1 2 2 1)	(2 2 2 1)
1	Y	2	{(1407, 14), (1466, 129), (1536, 8), (1536, 74) }	(2 2 1 1)	(2 1 1 1)
1	Y	3	{(32, 2452), (48, 2320), (40, 2376), (0, 2304)}	(1 2 2 2)	(1 2 1 2)
1	Y	4	{(32, 2456), (48, 2320), (40, 2376), (0, 2304)}	(2 1 1 1)	(1 2 1 2)

focus is on moving away from benchmark-driven empirical characterization of machine learning implementations towards developing a science of test for intelligent systems.

ACKNOWLEDGMENT

We acknowledge support from the Department of Homeland Security through agreement 43WV75701. The authors at the University of Central Florida acknowledge support from the Oak Ridge National Laboratory / UT Battelle, LLC under contract #4000136396, the National Science Foundation's Software & Hardware Foundation (#1422257) and Exploiting Parallelism & Scalability (#1438989) projects, and the XSEDE Extreme Science and Engineering Discovery Environment through Project# TG-CIE150022. FH has been supported by a Graduate Research Excellence Fellowship from the University of Central Florida. The authors would like to especially thank all anonymous reviewers for their insightful comments and suggestions.

REFERENCES

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[2] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Theory in practice for system design and verification," *ACM Siglog News*, vol. 2, no. 1, pp. 46–51, 2015.

[3] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre, "Software verification with blast," in *Model Checking Software*. Springer, 2003, pp. 235–239.

[4] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[5] C. Barrett, P. Fontaine, and C. Tinelli, "The smt-lib standardversion 2.5," 2010.

[6] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani, "Probabilistic programming," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 167–181.

[7] C. Courcoubetis and M. Yannakakis, "Verifying temporal properties of finite-state probabilistic programs," in *Foundations of Computer Science, 1988., 29th Annual Symposium on*. IEEE, 1988, pp. 338–345.

[8] D. Kozen, "Semantics of probabilistic programs," *Journal of Computer and System Sciences*, vol. 22, no. 3, pp. 328–350, 1981.

[9] C. Flanagan and S. Qadeer, "Predicate abstraction for software verification," in *ACM SIGPLAN Notices*, vol. 37, no. 1. ACM, 2002, pp. 191–202.

[10] E. W. Weisstein, " "Hypothesis Testing." – From MathWorld–A Wolfram Web Resource." mathworld.wolfram.com/HypothesisTesting.html, accessed: 2015-11-26.

[11] "OpenCV: Open Sourced Computer Vision," http://opencv.org/, accessed: 2015-11-26.

[12] R. B. Grosse and D. K. Duvenaud, "Testing mcmc code," *arXiv preprint arXiv:1412.5218*, 2014.

[13] M. Pacula. (2011, February) Unit-testing statistical software. [Online]. Available: http://blog.mpacula.com/2011/02/17/unit-testing-statistical-software/

[14] L. L. Pullum and A. Ramanathan, "Quantitative Approaches to Verify and Validate Anomaly Detection Algorithms," Oak Ridge National Laboratory, Tech. Rep., 2015.

[15] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.

[16] Z. Q. Zhou, D. Huang, T. Tse, Z. Yang, H. Huang, and T. Chen, "Metamorphic testing and its applications," in *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*, 2004, pp. 346–351.

[17] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins, and D. Powell, "Fault injection for dependability validation: A methodology and some applications," *Software Engineering, IEEE Transactions on*, vol. 16, no. 2, pp. 166–182, 1990.

[18] A. Gupta, "Formal hardware verification methods: A survey," in *Computer-Aided Verification*. Springer, 1993, pp. 5–92.

[19] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[20] G. J. Holzmann, *The SPIN model checker: Primer and reference manual*. Addison-Wesley Reading, 2004, vol. 1003.

[21] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal methods for performance evaluation*. Springer, 2007, pp. 220–270.

[22] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L.-J. Hwang, "Symbolic model checking: 10 20 states and beyond," in *Logic in Computer Science, 1990. LICS'90, Proceedings., Fifth Annual IEEE Symposium on e*. IEEE, 1990, pp. 428–439.

[23] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification*. Springer, 2010, pp. 122–135.

[24] S. K. Jha and A. Ramanathan, "Quantifying uncertainty in epidemiological models," in *BioMedical Computing (BioMedCom), 2012 ASE/IEEE International Conference on*. IEEE, 2012, pp. 80–85.

[25] O. P. Le Maître and O. M. Knio, *Introduction: Uncertainty Quantification and Propagation*. Springer, 2010.

[26] A. Wald, *Sequential analysis*. Courier Corporation, 1973.

[27] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random structures and algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[28] M. G. Bulmer, *Principles of statistics*. Courier Corporation, 2012.

[29] C. Fernández and M. F. Steel, "Multivariate student-t regression models: Pitfalls and inference," *Biometrika*, vol. 86, no. 1, pp. 153–167, 1999.

[30] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.