Research Statement

Kevin Matulef

December 15, 2013

1 Introduction

How can we design algorithms to extract the maximum amount of information out of the smallest amount of data? What is the tradeoff between the amount of data we need, and the quality of our conclusions? And how is this tradeoff affected by the structure of the data itself?

These are the questions my research has set out to solve. Of course, the problem of making decisions based on partial information is not new. Indeed, mathematicians have been studying such problems for hundreds of years, since the invention of formal probability theory (at least). But recently, these questions have taken on new forms. This has happened for two reasons. First, the technology for obtaining data has changed. Traditional statistical questions were motivated by the case where data was expensive or time-consuming to obtain. But now we often have the opposite problem: data is cheap to obtain, but expensive to process in its entirety. The ubiquity of computational devices, all connected to a network backed by cheap storage devices, has resulted in vast troves of stored information. But it is inefficient to crunch petabytes of data merely to get at small statistical facts, such as computing an average. In both cases— whether data is too expensive to obtain, or too expensive to process in its entirety— it has become crucial to draw conclusions using only a *portion* of the data.

A second reason for taking a new look at the old problems of partial data analysis is that the modern data sets we want to analyze often have extra structure; perhaps they represent edges of a large graph, or the inputs/outputs of a discrete function. The existence of such structure calls for the use of algorithmic tools, in addition to those of traditional probability theory. Indeed, we would like a theory of algorithms that do not just scale linearly with the input size, but scale *sub*-linearly, and still yield fast, approximate answers.

The challenge of analyzing massive data sets in sublinear time is an important but seldom discussed aspect of the so-called "Big Data" problem. Though the phrase "Big Data" is used to mean many things in the popular press, to me it describes a collection of distinct but interrelated challenges in different areas of Computer Science. On one hand is the challenge of building large distributed systems, which are able to ingest large amounts of information, and process data in parallel. On the other hand is the algorithmic challenge of analyzing large data sets in a mathematically formal way, to extract useful bits of information. Most of my previous research has focused on the second aspect— for instance, I have developed sublinear algorithms that can serve as precursors to traditional machine learning algorithms. But recently, I spent a year working as an engineer at MongoDB Inc., which has given me perspective on the first aspect as well. Ultimately, I believe cooperation is needed between researchers in both areas, and my background as both a theorist and an engineer has positioned me to bridge the gap. Indeed, as I will discuss later in this proposal, I am currently working on developing a new framework for sublinear algorithms that more closely models how real-world data is stored.

2 Previous Work

Most of my previous work has focused on a particular class of sublinear algorithms for decision problems, called *Property Testers*. At a high level, the goal of a property testing algorithm is simply to decide whether its input has some property or not. However, deciding this exactly is usually impossible in sublinear time. For instance, the input might be a list of numbers, and the algorithm would like to decide whether the list is sorted or not. Determining this exactly requires looking at the entire list, since it might contain just a single element that is out-of-order (and, in the worst case, it could be the last element the algorithm sees). Thus, we relax the problem, and only require that property testers provide an *approximate* answer. For instance, to test sortedness, we ask that the tester outputs "YES" on all sorted lists, and "NO" on all lists that are ϵ-far from sorted (meaning you would need to remove an ϵ fraction of the list elements for the remaining values to be sorted). The testing algorithm should do this while minimizing the number of values it queries in the list. More formally, we can think of a list of n real numbers as being represented by a function $f : [n] \to \mathbb{R}$, and the testing algorithm can query f, but would like to do so as few times as possible.

By only requiring that property testers distinguish between inputs that have the property versus those that are far, we can develop algorithms that make surprisingly few queries. (For instance, to test whether a list is sorted versus ϵ-far, prior work has shown that $\Theta(\log n/\epsilon)$ queries are both necessary and sufficient [EKK$^+$00, Fis04].) Over the last 15 years, property testing has emerged as an important area of theoretical computer science, and researchers have developed algorithms for testing a multitude of different properties (see, e.g., the surveys of [Gol11, Ron08, RS11]).

My own contributions to property testing are threefold: 1) I helped develop *new algorithmic techniques* for testing a broad range of important properties that were not previously known to be testable, 2) I helped develop a *new lower bound technique* for analyzing the complexity of many different testing problems, 3) I helped develop a *new model* for property testing that more closely reflects how some data sets are stored in practice.

2.1 New Algorithmic Techniques for Testing Function Properties

My research in graduate school was on developing new algorithms for testing different properties of functions. In particular, I focused on properties that are relevant from a machine learning perspective.

In a typical machine learning setup, one usually assumes that an unknown function f comes from a particular class of functions, and the goal is to learn an approximation to f, using a small number of example pairs of the form $(x, f(x))$. For instance, one might assume that f is a Boolean function defined by a halfspace (i.e., on a given input, it decides whether to output 1 or -1 depending upon which side of a hyperplane it lies). The goal of the learning algorithm is then to find an approximation to f by finding the separating hyperplane.

However, in real life, one usually does not know whether f has the form of a halfspace, or the form of a small decision tree, or anything for that matter. Moreover, any algorithm for learning an approximation to f must see a number of examples that depends on the number of inputs to f, which can often be quite large.

Unlike a learning algorithm, the goal of testing is simply to *distinguish* whether the function f is a member of some class (e.g. halfspaces), or "far" from all such functions. This can be viewed as a relaxation of the learning problem, because, loosely, a learning algorithm can always be used to test, but not the other way around. Indeed, if testing membership in a given function class requires fewer queries than learning that class, the testing algorithm can be used as an inexpensive

way to check whether a function has a given form, before bothering to run the associated learning algorithm. Much of my earlier research focused on identifying classes for which this was the case.

Testing Halfspaces

Perhaps the most well-studied class of functions in machine learning is the class of halfspaces, or functions of the form $f(x) = \text{sgn}(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \theta)$. Over the last forty years, from the perceptron algorithm to support vector machines, countless papers have been published on the topic of learning halfspaces.

In [MORS09], my co-authors and I set out to determine whether we could develop more efficient algorithms for the problem of *testing* halfspaces. In our paper, we gave the first algorithm for testing whether a Boolean function of the form $f : \{-1, 1\}^n \to \{-1, 1\}$ is a halfspace (versus ϵ-far from one) using $\text{poly}(1/\epsilon)$ queries. Note this number is *independent* of n, the number of inputs to f, thus showing that the testing problem can be solved much more efficiently than the learning problem (which requires $\Omega(n)$ queries).

In addition to being a surprising algorithmic result, our proof leveraged some sophisticated mathematical techniques, proving structural lemmas about halfspaces. These techniques have already proven useful for other applications, for instance in a line of recent work on the problem of *learning* halfspaces from their low-degree Fourier coefficients [OS11, DDFS12].

Testing Concise Representation Classes

In addition to halfspaces, there are many other important classes of functions in learning theory, such as the class of functions representable by small decision trees, or small DNF formulas.

In [DLM$^+$07], my co-authors and I developed a general new technique, which can be used to test membership in many classes of functions with "concise" representations. These classes include s-sparse polynomials, size-s decision trees, and s-term DNF formulas (the latter resolved an open question of [PRS02]). In each case, the query complexity of our algorithm was polynomial in s and ϵ, in contrast to the query complexity of the best learning algorithms for these classes, which must necessarily depend on n. We achieved our result by combining techniques from property testing (specifically the result of [FKR$^+$04] on testing juntas) and those from computational learning theory.

Our technique for testing concise representation classes inspired a variety of followup work by other authors (see, e.g., [AB10, CGSM11b, CGSM11a]). And in [DLM$^+$08], my co-authors and I improved our algorithm for s-sparse polynomials, giving a new algorithm that not only uses a small number of queries (polynomial in s and ϵ), but also exponentially improves the running time.

2.2 A New Lower Bound Technique

The flip side of working on algorithmic upper bounds is, of course, trying to prove lower bounds. As a postdoc at IIIS at Tsinghua University, I devoted a significant amount of effort towards this endeavor.

Historically, lower bounds for property testers have been difficult to prove, partly owing to the dearth of available techniques. But in [BBM12], my co-authors and I gave a new technique, based on the method of communication complexity. Using our technique, we were to able to resolve multiple open questions. For instance, we gave tight lower bounds for testing the class of k-linear functions (an open problem of [Gol10]), and for testing the class of monotone functions of the form $f : \{0, 1\} \to \mathbb{R}$. In addition to our new lower bounds, we also gave simpler proofs of several known lower bounds.

The communication complexity technique, developed over three decades ago by Andrew Yao [Yao79], involves analyzing the number of bits that two (or more) parties must communicate in order to compute some joint function of their inputs. Communication complexity is well-studied, and many lower bounds for communication problems are already known. Thus, by reducing from such problems, one can use the known results to prove lower bounds for other problems. Communication complexity had previously been used to prove lower bounds in areas such as streaming algorithms and circuits, but our paper was the first to realize a connection to property testing. As the strength of our results showed, this connection is quite deep.

Our [BBM12] result was one of the few chosen from the CCC conference to appear in a special issue of *Computational Complexity*, and our technique has already been used extensively in followup work by us and other researchers [Gol13, BRY13, BMW11].

2.3 A New Model of Property Testing on Linked Lists

A fundamental assumption of most property testing algorithms is that they can access their input by querying it arbitrarily. For instance, when testing functions, one typically assumes that the algorithm can query the function on any point in its domain. Unfortunately, many data sets cannot be accessed like this. Instead, they must be accessed in a way that respects the underlying data structure in which they are stored.

In recently submitted work [AMW13], my co-authors and I take a step towards modeling property testers with more realistic access constraints. We consider the classic problem of testing *sortedness*, mentioned earlier in this proposal. In the well-known version of this problem, the input is modeled as a function $f : [n] \to \mathbb{R}$, and the testing algorithm must minimize the number of queries it makes to f. As mentioned, in this setting it is known that $\Theta(\log n/\epsilon)$ queries are both necessary and sufficient [EKK+00, Fis04].

The classic version of testing sortedness models the situations where the input is stored as a large array, since querying f is akin to accessing an array element. But what if the input is stored as a linked list instead? In our work, we define a model for property testing on linked lists. In the linked list model, a testing algorithm can access a random element of the list, or walk to elements that are adjacent to those it has already seen, but cannot access a specific element at some arbitrary rank. We manage to prove that even though this model is more restrictive than the usual model, a non-trivial testing algorithm can still be achieved. In fact, we prove that in our model, $\Theta(n^{1/3}/\epsilon)$ queries are both necessary and sufficient.

3 Future Plans

It is always easier to discuss what one has done than anticipate what one will do. Thus, while I cannot say with certainty exactly what I will work on next, I can describe some of my research goals, both in the short-term and the long-term.

In the short-term, there are still plenty of research problems related to the theoretical foundations of sublinear algorithms that I think are worthwhile to study for the mathematical insight they provide. Some particular projects that I am currently working on are the following:

Testing on Data Structures

Our recent work [AMW13] on property testing on linked lists is just the tip of the iceberg, as it opens up a host of interesting questions about testing on different kinds of data structures. A natural extension to the result of [AMW13] would be to consider testing "sortedness" on trees,

or even arbitrary partially ordered sets, instead of just lists. On trees, for instance, an algorithm would be allowed to jump to a random place in the tree, or walk to parents/children, but not to specific nodes. The goal, then, would be to determine whether the values stored at each node are monotone increasing from the children to the parents. For binary trees, this actually corresponds to the *heap* property- i.e. such an algorithm could be used to test whether a tree has a heap structure, or whether it has been corrupted and is "far" from a heap.

Lower Bounds via Information Theory

The lower bound method we developed in [BBM12] works by reducing communication problems with known lower bounds to testing problems. As previously mentioned, this method has proven to be surprisingly powerful. However there are still some testing problems for which we have not been able to find clever reductions. This raises the question of whether the reductions are necessary, or whether we might apply techniques from information theory (which are behind some of the communication lower bounds themselves) directly to the property testing problems. My colleagues and I have already begun investigating this approach, and are hopeful that it will yield new lower bounds.

In the long-term, my goal is to make sublinear algorithms more applicable. Currently, the field of property testing is full of mathematically beautiful theorems, but with few exceptions, it is divorced from practical application. There are multiple reasons for this. First, the property testing model is often unrealistic in practice. For instance, most algorithms do not have arbitrary query access to their input, but instead are limited in their access capabilities. As discussed earlier, part of my motivation in the recent work of [AMW13] was to take a step towards addressing this.

A second reason why property testing has yet to find many practical applications is simply that researchers in the field have not made much effort to reach out to practitioners. Historically, the field was founded by complexity theorists, who had more interaction with other mathematicians than with engineers. But my own background is different. As an engineer at MongoDB Inc., I got my hands dirty, doing everything from programming MongoDB's hashed sharding framework, to dealing with customer questions. The time I spent there gave me industry contacts, and a desire to address some of the technical challenges they faced.

At MongoDB, I witnessed several cases where the engineers were not using the asymptotically optimal algorithm or data structure to solve the problem at hand. This was not due to ignorance or ineptitude. Rather, it was a calculated tradeoff, since the optimal method was either more complicated to implement, or offered little advantage on real-world inputs, which had extra structure and did not look like "worst-case" inputs. To me, this presents an excellent research opportunity. Can sublinear algorithms be used as a tool for algorithmic optimization? By first testing whether the data has a given form, can one then use an algorithm or data structure with provable performance guarantees on data of that form? Such an approach was already used by researchers to optimize certain database queries in [BMKF$^+$11]. In the future, I plan to leverage both my theoretical expertise and my engineering experience to identify other areas where sublinear algorithms can have large practical impact.

References

[AB10] Noga Alon and Eric Blais. Testing boolean function isomorphism. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 394–405. Springer, 2010.

[AMW13] Peyman Afshani, Kevin Matulef, and Bryan Wilkinson. Property testing on linked lists. *Manuscript*, 2013.

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

[BMKF+11] Sagi Ben-Moshe, Yaron Kanza, Eldar Fischer, Arie Matsliah, Mani Fischer, and Carl Staelin. Detecting and exploiting near-sortedness for efficient relational query evaluation. In *Proceedings of the 14th International Conference on Database Theory*, pages 256–267. ACM, 2011.

[BMW11] Joshua Brody, Kevin Matulef, and Chenggang Wu. Lower bounds for testing computability by small width obdds. In Mitsunori Ogihara and Jun Tarui, editors, *TAMC*, volume 6648 of *Lecture Notes in Computer Science*, pages 320–331. Springer, 2011.

[BRY13] Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions on hypergrid domains. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 20, page 36, 2013.

[CGSM11a] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *Automata, Languages and Programming*, pages 545–556. Springer, 2011.

[CGSM11b] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1683–1702. SIAM, 2011.

[DDFS12] Anindya De, Ilias Diakonikolas, Vitaly Feldman, and Rocco A Servedio. Nearly optimal solutions for the chow parameters problem and low-weight approximation of halfspaces. In *Proceedings of the 44th symposium on Theory of Computing*, pages 729–746. ACM, 2012.

[DLM+07] Ilias Diakonikolas, Homin Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco Servedio, and Andrew Wan. Testing for concise representations. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 549–558, 2007.

[DLM+08] Ilias Diakonikolas, Homin K Lee, Kevin Matulef, Rocco A Servedio, and Andrew Wan. Efficiently testing sparse gf (2) polynomials. In *Automata, Languages and Programming*, pages 502–514. Springer, 2008.

[EKK+00] Funda Ergun, Sampath Kannan, Ravi Kumar, Ronitt Rubenfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60:717–751, 2000.

[Fis04] Eldar Fischer. On the strength of comparisons in property testing. *Info. and Comput.*, 189(1):107–116, 2004.

[FKR+04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *J. Comput. Syst. Sci.*, 68:753–787, 2004.

[Gol10] Oded Goldreich. On testing computability by small width OBDDs. In *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 574–587, 2010.

[Gol11] Oded Goldreich. Introduction to testing graph properties. In Oded Goldreich, editor, *Property Testing*, volume 6390 of *Lecture Notes in Computer Science*, pages 105–141. Springer Berlin Heidelberg, 2011.

[Gol13] Oded Goldreich. On the communication complexity methodology for proving lower bounds on the query complexity of property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:73, 2013.

[MORS09] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco Servedio. Testing {-1,1}-weight halfspaces. In *Proc. 13th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2009.

[OS11] Ryan O'Donnell and Rocco A Servedio. The chow parameters problem. *SIAM Journal on Computing*, 40(1):165–199, 2011.

[PRS02] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM J. Disc. Math.*, 16(1):20–46, 2002.

[Ron08] Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

[RS11] Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM J. Disc. Math.*, 25(4):1562–1588, 2011.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213. ACM, 1979.