DOI:10.1145/3528087

Michael Mitzenmacher and Sergei Vassilvitskii

Viewpoint Algorithms with Predictions

Seeking a new approach that goes beyond worst-case analysis.

THE THEORETICAL STUDY of algorithms and data structures has been bolstered by worst-case analysis, where we prove bounds on the running time, space, approximation ratio, competitive ratio, or other measure that holds even in the worst case. Worst-case analysis has proven invaluable for understanding aspects of both the complexity and practicality of algorithms, providing useful features like the ability to use algorithms as building blocks and subroutines with a clear picture of the worst-case performance. More and more, however, the limitations of worst-case analysis become apparent and create new challenges. In practice, we often do not face worst-case scenarios, and the question arises of how we can tune our algorithms to work even better on the kinds of instances we are likely to see, while ideally keeping a rigorous formal framework similar to what we have developed through worst-case analysis.

A key issue is how we can define the subset of "instances we are likely to see." Here we look at a recent trend in research that draws on machine learning to answer this question. Machine learning is fundamentally about generalizing and predicting from small sets of examples, and so we model additional information about our algorithm's input as a "prediction" about our problem instance to guide and hopefully improve our algorithm. Of course, while ML performance has made tremendous strides in a short amount of time, ML predictions can be error-prone, with unexpected results, so we must take care in how much our algorithms trust their predictors. Also, while we suggest ML-based predictors, predictions really can come from anywhere, and simple predictors may not need sophisticated machine learning techniques. For example, just as yesterday's weather may be a good predictor of today's weather, if we are given a sequence of similar problems to solve, the solution from the last instance may be a good guide for the next.

What we want, then, is merely the best of both worlds. We seek algorithms augmented with predictions that are:

- ▸ Consistent: when the predictions are good, they are near-optimal on a per instance basis;
- ▸ Robust: when the predictions are bad, they are near-optimal on a worst-case basis;
- ▸ Smooth: the algorithm interpolates gracefully between the robust and consistent settings; and
- ▸ Learnable: we can learn whatever we are trying to predict with sufficiently few examples.

Our goal is a new approach that goes beyond worst-case analysis.[14] We identify the part of the problem space that a deployed algorithm is seeing and automatically tune its performance accordingly.

As a natural starting example, let us consider binary search with the addition of predictions. When looking for an element in a large sorted array, classical binary search compares the target with the middle element and then recurses on the appropriate half (see Figure 1). Consider, however, how we find a book in a bookstore or library. If we are looking for a novel by Isaac Asimov, we start searching near the beginning of the shelf, and then look around, iteratively doubling our search radius if our initial guess was far off (see Figure 2). We can make this precise to show that there is an algorithm with running time logarithmic in the error of our initial guess (measured by how far off we are from the correct location), as opposed to being logarithmic in the number of elements in the array, which is the standard result for binary search. Since the error is no larger than the size of the array, we obtain an algorithm that is consistent (small errors allow us to find the element in constant time) and robust (large errors recover the classical $O(\log n)$ result, albeit with a larger constant factor).

Many readers may notice this is a variation on the idea of interpolation search, using only a predicted starting point. (Interpolation search uses the data to estimate the next comparison point, instead of always picking the middle as in binary search.) With this view, algorithms with predictions have been in the air for some time, and the ML explosion has simply provided motivation to both expand the idea and develop richer formalizations.

A recent success along these lines formalizes the idea of 'warm start.' When repeatedly solving similar optimization problems, practitioners often don't start from scratch each time, but instead start searching near a previous solution. Dinitz et al.[4] analyze the performance gains of treating such a solution as a prediction in the context of min cost perfect matchings. In their setting, one solves a number of problems on the same graph, but with different edge weights for each instance, where the edge weights may, for example, come from a distribution. They show that given a prediction for the dual solution of a corresponding linear program, they can compute a feasible dual solution from it, improving the overall running time and expanding upon the "use the solution from yesterday" heuristic.

Predictions have also been suggested as a means for reducing space usage for several data structures, for example notably in the seminal work of Kraska et al.[7] for learned indices. As an example of how predictions can save space, we first explain the later work of Hsu et al.[6] on data structures for frequency estimation that use learning.

Frequency estimation algorithms are used to approximately count things, such as the number of packets a router sees sent from each IP address. Since it can be expensive in both space and time to keep a separate counter for each address, estimation algorithms use techniques such as hashing each address into a table of shared counters (usually hashing each address into several locations for robustness), and then deriving an estimate when queried for an IP address from its associated counters. The largest count estimate errors occur when an address with a small count hashes to the same locations as addresses with a large count, as it then appears that the address should itself have a high count. If we somehow knew the addresses with large counts ahead of time, we could assign them their own counters and handle them separately from the sketch, avoiding such large errors and obtaining better frequency estimation with smaller overall space. The paper by Hsu et al.[6] introduces the idea of using machine learning to predict which objects (in this example, IP addresses) have large counts, and separate them out in this way. They prove bounds for specific cases and demonstrate empirically both that high-count elements are predictable and that using such predictions can lead to improved practical performance.

As another example of how predictions can save space, Kraska et al.[7] propose a framework for learned Bloom filters. Bloom filters are compressed data structures for set membership; for a set X of keys, a Bloom filter correctly returns yes for any x that is truly in X, but may give a false positive for keys not in the set. Bloom filters have a space-accuracy trade-off, where more space allows for fewer false positives. The work of Kraska et al.[7] suggests that if a set can be learned, that is, a predic-

Figure 1. The execution of traditional binary search.

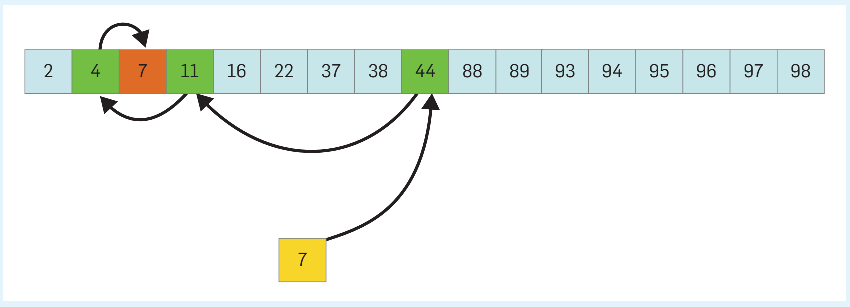
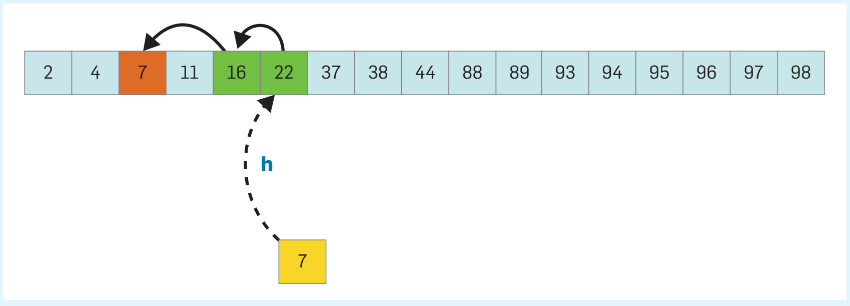


Figure 2. The execution of binary search, starting with a prediction.



tor can imperfectly predict whether an element is or is not in the set, that can be used to derive a learned Bloom filter that combines the predictor with a standard Bloom filter in a way that improves the space-accuracy trade-off. We leave the details of the various improved learned Bloom filters to the relevant papers.[7,9,17]

Perhaps unsurprisingly, one area where using predictions is having a tremendous impact is for online algorithms, where the algorithm responds to an incoming data stream and the future is unknown. The theoretical framework of competitive analysis considers the worst-case ratio between the performance of an online algorithm and the optimal algorithm as a measure, so a "two-competitive" algorithm is always within a factor of two of optimal. Coping with the worst-case possible future is often difficult, and thus taking advantage of predictions in this setting is often quite powerful. For example, some recent results consider scheduling problems. Jobs arrive over time at a single server and have to be scheduled; the cost for each job is the time between when it arrives and when it finishes, and one wants to minimize the total cost over all jobs. If a job's required processing time is known on arrival, then scheduling by Shortest Remaining Processing Time (SRPT) is optimal. But what if only estimates of job times are known? Recent work shows that if every job with true size s has an estimate between $[bs,as]$ for constants a,b with $0 < b < 1 < a$, there is an algorithm with competitive ratio $O((a/b)\log^2(a/b))$, even if the algorithm does not know a and b in advance. That is, one can achieve performance close to optimal, and the performance gracefully degrades with the estimate quality.[2] Scheduling with predictions has similarly been studied in the context of queueing theory, where the models have probabilistic assumptions, such as Poisson arrivals and independent and identically distributed service times. In this setting, when using estimates, SRPT can perform quite badly even when estimates are again bounded in $[bs, as]$ for a job of size s, but a variation of SRPT using estimates converges to the performance of SRPT with full information as a and b go to 1, and is within $O(a/b)$ of SRPT always, again without knowing a and b in advance.[15] Other work looking at the queueing setting has shown that even one bit of advice, predicting whether a job is short or long for some suitable notion of short or long, can greatly improve performance.[10] Several other online problems have been studied with predictions, including caching,[8] online clustering, and the historically fun and enlightening ski rental[13] and secretary problems.[1,5]

Predictions have also been suggested as a means for reducing space usage for several data structures.

It is worth noting there is also a great deal of recent work in the closely related area of data-driven algorithm design. At a high level, this area often studies the tuning of an algorithm's hyperparameters, such as the step-size in a gradient descent, or which of the many possible initializations for k-means clustering is best. (The survey by Balcan[3] provides a deep dive into this area.)

The research area of Algorithms with Predictions has really only just started, but it seems to be booming, as researchers reexamine classical algorithms and see where they can be improved when good predictions are available. This marriage of classical algorithms and data structures with machine learning may lead to significant improvements in systems down the road, providing benefits when good predictions are available (as they seem to be in the real world) but also limiting performance downsides when predictions go wrong (as, inevitably, also seems to happen in the real world).

For those interested in more technical detail, we have a short survey available,[11] and there are related recent workshops with talks online.[12,16] C

References

1. Antoniadis, A. et al. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, H. Larochelle et al., Eds. NeurIPS 2020 (Dec. 6–12 (virtual) 2020).
2. Azar, Y., Leonardi, S., and Touitou, N. Distortion-oblivious algorithms for minimizing flow time. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms*, SODA 2022, S. Naor and N. Buchbinder, Eds. (Jan. 9–12, 2022), 252–274.
3. Balcan, M. Data-driven algorithm design. CoRR abs/2011.07177 (2020).
4. Dinitz, M. et al. Faster matchings via learned duals. In Advances in Neural Information Processing Systems (2021), M. Ranzato, A. et al., Eds., vol. 34, Curran Associates, Inc., 10393–10406.
5. Dütting, P. et al. Secretaries with advice. In *EC '21: The 22nd ACM Conference on Economics and Computation*. P. Biró, S. Chawla, and F. Echenique, Eds., Budapest, Hungary, July 18–23, 2021t al. (2021), ACM, 409–429.
6. Hsu, C. et al. Learning-based frequency estimation algorithms. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, (New Orleans, LA, USA, May 6–9, 2019); OpenReview.net.
7. Kraska, T. et al. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*. G. Das, C.M. Jermaine, and P.A. Bernstein, Eds. SIGMOD Conference 2018 (Houston, TX, USA, June 10–15, 2018), 489–504.
8. Lykouris, T., and Vassilvitskii, S. Competitive caching with machine learned advice. *J. ACM 68*, 4 (2021).
9. Mitzenmacher, M. A model for learned BLOOM filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, NeurIPS 2018, S. Bengio et al., Eds. (Dec. 3–8, 2018, Montréal, Canada (2018), 462–471.
10. Mitzenmacher, M. Queues with small advice. In *Proceedings of the 2021 SIAM Conference on Applied and Computational Discrete Algorithms*, ACDA 2021, M. Bender, et al., Eds., (July 19–21, (virtual) 2021), 1–12.
11. Mitzenmacher, M., and Vassilvitskii, S. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, T. Roughgarden, Ed. Cambridge University Press, 2020, 646–662.
12. ML4A 2021—Machine Learning for Algorithms (July 2021); https://bit.ly/3wThaVs
13. Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*. S. Bengio et al., Eds. NeurIPS 2018 (Dec. 3–8, 2018), Montréal, Canada (2018), 9684–9693.
14. Roughgarden, T., Ed. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.
15. Scully, Z., Grosof, I., and Mitzenmacher, M. Uniform bounds for scheduling with job size estimates. In 13th Innovations in Theoretical Computer Science Conference, ITCS 2022, Jan.–Feb. 3, 2022, Berkeley, CA, USA (2022), M. Braverman, Ed., vol. 215 of LIPIcs, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, pp. 114:1–114:30.
16. STOC 2020 - Workshop 5: Algorithms with Predictions. https://bit.ly/3wThgwi
17. Vaidya, K. et al. Partitioned learned BLOOM filters. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021 (2021), OpenReview.net.

Michael Mitzenmacher (michaelm@eecs.harvard.edu) is the Thomas J. Watson, Sr. Professor of Computer Science at the Harvard School of Engineering and Applied Sciences, Cambridge, MA, USA.

Sergei Vassilvitskii (sergeiv@google.com) is a principal scientist at Google Research, New York, NY, USA.

Michael Mitzenmacher was supported in part by NSF grants CCF-2101140, CNS-2107078, and DMS-2023528.