Research Statement

Ashok Cutkosky

I am broadly interested in making machine learning more about engineering and less about intuition and guesswork. Although machine learning is widely and profitably deployed throughout industry, many crucial design decisions are driven primarily by trial and error rather than by principled understanding. High-level decisions (e.g. model architecture) as well as low-level decisions (e.g. learning rates) are typically made by trying out many options and then picking the "best" one via some specified metric. If something does not work, it may not be easy to tell if the fault is due to a programming bug, or one of these decisions (let alone which one). I want the design of machine learning systems to be closer to traditional algorithm design, in which the foundations are solid enough that when one writes a program for some task, one can provide a convincing argument that the program will, indeed, accomplish that task. I believe that developing these foundations will dramatically accelerate the rate of existing machine learning application development.

My prior work in this area has focused on developing *adaptive optimization algorithms*. Optimization is one of the basic building blocks of machine learning: whenever one trains a model (such as a neural network, an SVM, a decision tree, etc.) to perform well on some data, the training algorithm is nearly always optimizing model parameters (such as linear regression or neural network weights) to improve some objective (such as mean squared error). In optimization, the guesswork commonly employed in practical machine learning manifests as the need to tune some parameters of the algorithm (for example, the learning rate) in order to obtain better performance on a particular dataset. In contrast, an adaptive algorithm should automatically adjust to the data supplied to it, ideally performing nearly as well as an algorithm that was specifically tuned. This allows us to eliminate at least one source of the guess-and-check involved in machine learning. Adaptive algorithms can have a number of other benefits: they reduce the amount of compute resources used by eliminating the need to test different parameter settings, they can reduce overfitting caused by testing many different algorithm parameters on a dataset, and sometimes they can even exceed the performance of a tuned algorithm by adapting to more types of parameters than could be checked via manual tuning in any reasonable amount of time.

1 Adaptive Online Learning

In pursuit of finding these adaptive algorithms, I have focused on online learning, which is an elegant framework for analyzing both stochastic and streaming optimization problems. Online learning is a "game" played in T rounds. In each round, a learning algorithm outputs a vector w_t, suffers loss $\ell_t(w_t)$ for some function ℓ_t, and sees some feedback about the loss, such as $\nabla \ell_t(w_t)$. The goal is to suffer nearly the same total loss $\sum_{t=1}^T \ell_t(w_t)$ as a comparison point $w_\star$: $\sum_{t=1}^T \ell_t(w_\star)$. In order to make the problem tractable, we also assume that each loss ℓ_t is convex. Common practical problems such as stochastic optimization, learning with expert advice, and sequential investment are easily modeled in this framework. For example, to iteratively train a linear regression model on some dataset using online learning, set w_t to be the weight at the tth iterate and $\ell_t(w_t) = (x_t \cdot w_t - y_t)^2$ to be the squared error of the weights w_t on the tth element of some dataset, with feature x_t and label y_t. In fact, many of the iterative gradient-descent-like algorithms in common use today are online learning algorithms.

Although many algorithms that adapt to various properties of the losses had already been proposed and widely-adopted when I started working on this field [15, 16, 17], all of these methods maintained at least one parameter that needed to be tuned, intuitively representing either the distance $\|w_\star\|$ or the size of the gradients $\|\nabla \ell_t(w_t)\|$. I proved that in the worst-case, it is unfortunately impossible to adapt optimally to both parameters simultaneously [1], resolving a COLT open problem [18]. Despite this negative result, I later described a way to quantify the degree to which the losses/dataset exhibits worst-case properties, and provided a family of optimally adaptive algorithms [2]. This family characterizes a trade-off between the ability to achieve adaptivity when not in the worst-case, and a vulnerability to a potentially exponentially bad penalty otherwise. I earned the 2017 best student paper award at COLT for this result.

Much of my recent work in this field has produced new ways to analyze adaptive online learning algorithms through the use of *reductions*. These results have allowed me to significantly streamline analysis and develop more powerful algorithms by using seemingly weaker algorithms as subroutines. I believe this is a particularly attractive design methodology; the results are general and very easy to reuse in new contexts, and the proofs are usually simpler because we only work with limited assumptions about the base algorithms. I took a first step down this path in [3] by showing how one can adapt to properties of the second derivative of the losses using an algorithm that only looks at first-order information. Then, in [4], I provided a toolbox of powerful reductions that not only simplified many prior analysis, but also enabled several new results. For example, although many practical problems are high dimensional or have constraints, it is easier to design algorithms for 1-dimensional or unconstrained settings. My work resolves this tension by showing how to easily convert any 1-dimensional algorithm into a high-dimensional algorithm, and how to convert any unconstrained algorithm into a constrained algorithm. I have made further progress by building on top of these reductions; every new reduction accelerates progress because we can combine multiple reductions to produce algorithms that obtain many desired properties simultaneously. In this vein, I combined old and new reductions in [5] to obtain an algorithm that allows one to avoid my exponential penalty from [2] on most realistic datasets. Finally, in [6], I provided an extremely simple one-line construction that can combine any finite number of online algorithms to obtain a single algorithm whose performance will match that of whichever original algorithm happened to be the best. Although these results are very recent, their ease of use is such that they have already been applied by others to improve adaptivity of prior methods, or to obtain new differentially private optimization algorithms [19, 20, 21].

1.1 Future Directions

While there are still many interesting unanswered questions in adaptive online convex optimization, I am interested in expanding the scope of adaptivity beyond this setting. One next direction is the *bandit* scenario, which models black-box optimization by only providing the online algorithm with loss values $\ell_t(w_t)$ rather than gradients or other more detailed information. In fact, the last couple of years has already seen some interest in adaptive bandit algorithms [22, 23], and I am very excited to apply the new techniques that have worked well in the full information setting to this scenario.

2 Improving Efficiency

Adaptive optimization algorithms typically focus on minimizing the number of datapoints required to converge to the optimal solution (called the sample complexity), but in practice it is also important to make each iterate computationally efficient. To this end, I have worked on distributed optimization algorithms in [7] that can self-tune its learning rates to the geometry of the losses. In particular, the method is generic scheme to parallelize any sufficiently adaptive serial optimization algorithm. As a result, improvements in the simpler serial setting immediately translate to improvements in the distributed setting. Typical parallel optimization algorithms (e.g. batched SGD) incur a lot of sample complexity over non-distributed algorithms in practice, but amazingly we found that our method gained zero sample complexity even when running hundreds of threads in parallel.

In addition to parallelizing iterate computation, another way to increase the computational efficiency of an algorithm is to make each individual iterate faster to compute. A good candidate for this type of improvement is the family of so-called "full-matrix" or "preconditioned" optimization algorithms. These algorithms require very few iterates, but each iterate is very slow due to working with a large matrix. The tradeoff of fewer but more costly iterates is currently not favorable, and there has been much interest in unlocking the potential of these methods by speeding up the iterates [24, 25]. Prior work usually operates by compressing the matrix in a lossy way. This results in a compromise in terms of both number of iterates required and speed of iterates. In contrast, I developed an algorithm [8] that completely dispenses with the matrix, so that it is *just as fast* as a simple un-preconditioned gradient descent. Moreover, the number of iterates required is never worse than the un-preconditioned method, and sometimes even better than the preconditioned method, depending on certain properties of the data. Thus, this algorithm finds a new kind of tradeoff that strictly dominates un-preconditioned algorithms.

2.1 Future Directions

In the future, I am interested in examining the computational efficiency of non-convex optimization algorithms. This is of great practical relevance in the deep learning community, and specialized optimization algorithms have been designed to use very large batch sizes in order to parallelize the computations [26]. Despite their empirical performance, the theoretical advantages of these techniques are less clear, and so there is a clear opportunity to improve the theory, and in the process design improved practical algorithms.

3 Stochastic Optimization and Non-Convexity

To complement my work in online learning, I have also investigated stochastic optimization in convex and non-convex settings. This transition is especially natural in the convex case because any online algorithm can be converted into a stochastic optimization algorithm via the classic "online-to-batch conversion" [27]. I have recently improved this technique in [9], providing a modification that allows the method to be *anytime*. This means that the iterates w_t are such that the losses $\mathcal{L}(w_t)$ converge to the optimal loss value $\mathcal{L}(w_\star)$, an intuitively desirable property that the previous construction surprisingly lacked. This resolved another COLT open problem proposed later that year [28]. Further, in the same work, I also uncovered a connection to accelerated rates in deterministic smooth optimization. This allowed me to combine adaptive online learning and acceleration to derive the first algorithm to achieve optimal rates in both deterministic and stochastic settings without knowledge of any parameters such as smoothness constants or variance bounds.

In addition to the above perspective with arbitrary convex losses, in [10], we investigated the role of regularization in the generalization of kernel regression. In practice, it is often observed that no regularization is required to obtain good generalization behavior, even though theoretical generalization bounds usually require large amounts of regularization. We showed that zero regularization can provide much better generalization guarantees when it is possible to obtain very small error. This result holds for the kernel ridge regression setting, and I am interested in finding if similar statements can be made in more general and non-convex scenarios.

Moving from convex to non-convex optimization, I developed a reduction that recasts the problem of finding the best learning rate for non-convex stochastic gradient descent as itself a *convex* online learning problem. Thus, by applying convex methods, we can provably learn the optimal learning rate for non-convex problems [11]. I have also worked on *variance reduction* methods in non-convex optimization. Variance reduction currently provides the only known way to converge faster than stochastic gradient descent. While much of the work on stochastic gradient descent focuses on adaptivity, essentially none of the work on variance reduction does so. I have helped bridge this gap in [12]. I introduced an algorithm that adapts to the unknown amount of stochasticity in the problem, smoothly interpolating between faster rates in determininistic problems and the best-known stochastic rates when there is noise. Our techniques have already been extended to constrained and convex settings by others in [29].

3.1 Future Directions

Finally, the use of momentum is ubiquitous in deep learning, where it empirically accelerates stochastic gradient descent. Despite this, no current analysis shows that momentum in the typically applied form provides any benefit to stochastic gradient descent. I think this is a fascinating gap between theory and practice whose resolution may lead to new understanding of the dynamics of optimization algorithms on non-convex objectives.

4 Other Interests

While I expand my research in optimization theory, I intend to also delve more into other fields. For example, some of my earlier work in graduate school was related to developing efficient and expressive model architectures [13]. I am interested in investigating this further, with a view to identifying architectures that provably obtain good tradeoffs in terms of ease of optimization, ability to fit and generalize well on realistic data, and inference speed.

I also plan to investigate applications more tangential to machine learning. One such area is a recent trend of combining learning algorithms with classical algorithms design (e.g. [30, 31]). The idea is to augment traditional algorithms or data structures with some "advice" from a learning system. The algorithm should do better while the advice is good, but maintain reasonable worst-case performance when the advice is bad. These goals are extremely similar to those of adaptive optimization, and so I anticipate applying the insights I have gained in my prior work here.

One final goal of mine is to turn my work in machine learning into practical applications in biology and medicine. I have been intersted in this area for quite some time: my undergraduate thesis was on modeling genome topology, eventually contributing to [14], and I earned a Masters in Medicine at Stanford concurrently with my PhD. I am eager to merge my background in biology with my current work. For example, the current rapid rate of genomic data generation is making the ability to compress genomes more and more important. Since compression is closely related to learning, can learning techniques can be used to compress genomic data more efficiently? Could these same compression algorithms be used to help identify regions of interest (such as enhancers) in the genome? Since experiments in biology can take months to years, can we use online algorithms to help decide the order in which to perform experiments? Similarly, in directed evolution, one generates new proteins by random mutation of amino acid sequences. Can optimization theory help guide this randomness? One of the biggest appeals of academia over industry positions is the easier ability branch out into these new areas, and I am very much looking forward to doing so!

References By Applicant

[1] Ashok Cutkosky and Kwabena A Boahen. "Online Convex Optimization with Unconstrained Domains and Losses". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[2] Ashok Cutkosky and Kwabena Boahen. "Online Learning Without Prior Information". In: *Conference on Learning Theory*. 2017, pp. 643–677.

[3] Ashok Cutkosky and Kwabena A Boahen. "Stochastic and adversarial online learning without hyperparameters". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5059–5067.

[4] Ashok Cutkosky and Francesco Orabona. "Black-Box Reductions for Parameter-free Online Learning in Banach Spaces". In: *Conference On Learning Theory*. 2018, pp. 1493–1529.

[5] Ashok Cutkosky. "Artificial Constraints and Hints for Unbounded Online Learning". In: *Proceedings of the Thirty-Second Conference on Learning Theory*. 2019, pp. 874–894.

[6] Ashok Cutkosky. "Combining Online Learning Guarantees". In: *Proceedings of the Thirty-Second Conference on Learning Theory*. 2019, pp. 895–913.

[7] Ashok Cutkosky and Róbert Busa-Fekete. "Distributed stochastic optimization via adaptive SGD". In: *Advances in Neural Information Processing Systems*. 2018, pp. 1910–1919.

[8] Ashok Cutkosky and Tamas Sarlos. "Matrix-Free Preconditioning in Online Learning". In: *International Conference on Machine Learning*. 2019, pp. 1455–1464.

[9] Ashok Cutkosky. "Anytime Online-to-Batch, Optimism and Acceleration". In: *International Conference on Machine Learning*. 2019, pp. 1446–1454.

[10] Kwang-Sung Jun, Ashok Cutkosky, and Francesco Orabona. "Kernel Truncated Randomized Ridge Regression: Optimal Rates and Low Noise Acceleration". In: *Advances in Neural Information Processing Systems*. 2019.

[11] Zhenxun Zhuang, Ashok Cutkosky, and Francesco Orabona. "Surrogate Losses for Online Learning of Stepsizes in Stochastic Non-Convex Optimization". In: *International Conference on Machine Learning*. 2019, pp. 7664–7672.

[12] Ashok Cutkosky and Francesco Orabona. "Momentum-Based Variance Reduction in Non-convex SGD". In: *Advances in Neural Information Processing Systems*. 2019.

[13] Ashok Cutkosky and Kwabena Boahen. "Bloom Features". In: *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. 2015, pp. 547–552.

[14] Adrian L Sanborn et al. "Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes". In: *Proceedings of the National Academy of Sciences* 112.47 (2015), E6456–E6465.

Other References

[15] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Conference on Learning Theory*. 2010.

[16] H. Brendan McMahan and Matthew Streeter. "Adaptive Bound Optimization for Online Convex Optimization". In: *Proceedings of the 23rd Annual Conference on Learning Theory*. 2010.

[17] Francesco Orabona. "Simultaneous model selection and optimization through parameter-free stochastic learning". In: *Advances in Neural Information Processing Systems*. 2014, pp. 1116–1124.

[18] Francesco Orabona and Dávid Pál. "Open Problem: Parameter-Free and Scale-Free Online Algorithms". In: *Conference on Learning Theory*. 2016.

[19] Zakaria Mhammedi, Wouter M Koolen, and Tim Van Erven. "Lipschitz Adaptivity with Multiple Learning Rates in Online Learning". In: *Conference on Learning Theory*. 2019, pp. 2490–2511.

[20] Kwang-Sung Jun and Francesco Orabona. "Parameter-Free Online Convex Optimization with Sub-Exponential Noise". In: *Conference on Learning Theory*. 2019, pp. 1802–1823.

[21] Dirk van der Hoeven. "User-Specified Local Differential Privacy in Unconstrained Adaptive Online Learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 14080–14089.

[22] Chen-Yu Wei and Haipeng Luo. "More Adaptive Algorithms for Adversarial Bandits". In: *Conference On Learning Theory*. 2018, pp. 1263–1291.

[23] Sébastien Bubeck et al. “Improved Path-length Regret Bounds for Bandits”. In: *Conference on Learning Theory*. 2019, pp. 508–528.

[24] Haipeng Luo et al. “Efficient second order online learning by sketching”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 902–910.

[25] Naman Agarwal et al. “The case for full-matrix adaptive regularization”. In: *arXiv preprint arXiv:1806.02958* (2018).

[26] Yang You et al. “Large batch optimization for deep learning: Training bert in 76 minutes”. In: *arXiv preprint arXiv:1904.00962* (2019).

[27] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. “On the generalization ability of on-line learning algorithms”. In: *Information Theory, IEEE Transactions on* 50.9 (2004), pp. 2050–2057.

[28] Rong Ge et al. “Open Problem: Do Good Algorithms Necessarily Query Bad Points?” In: *Conference on Learning Theory*. 2019, pp. 3190–3193.

[29] Mingrui Zhang et al. “One Sample Stochastic Frank-Wolfe”. In: *arXiv preprint arXiv:1910.04322* (2019).

[30] Michael Mitzenmacher. “A model for learned bloom filters and optimizing by sandwiching”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 464–473.

[31] Sreenivas Gollapudi and Debmalya Panigrahi. “Online Algorithms for Rent-Or-Buy with Expert Advice”. In: *International Conference on Machine Learning*. 2019, pp. 2319–2327.