Optimization in Theory and Practice

Stephen Wright (UW-Madison)

U.Penn, April, 2024

Outline

How theoretical analysis of optimization algorithms relates to their practical performance.

Theory and practice tend to drive / motivate each other — often with lags and gaps. Varies widely across different problem classes.

- Theory and Practice: Intro
- Complexity in Nonlinear Optimization
    - History
    - Optimization problems from Machine Learning
    - Nonconvex Nonlinear Optimization
- Theory and Practice in Linear Programming (if time allows)

A personal perspective! Your mileage may vary...

Continuous Optimization Problems

Unconstrained Minimization:

$$\min_x f(x), \quad \text{where } f : \mathbb{R}^n \to \mathbb{R} \text{ is smooth.}$$

Constrained Optimization ("Nonlinear Programming / NLP"):

$$\min_x f(x) \text{ subject to } \; c(x) = 0, \; d(x) \geq 0,$$

$c : \mathbb{R}^n \to \mathbb{R}^p$ and $d : \mathbb{R}^n \to \mathbb{R}^m$ are smooth vector functions (constraints).

Many special cases have been studied, e.g.

- f, c, and d all linear (linear programming)
- f quadratic, c and d linear (quadratic programming)
- constraints $c(x)$ and $d(x)$ linear functions;
- c null, $d(x) = x$ (bound-constrained)
- d null (equality constrained)

Linear Programming (LP)

A very important special case is linear programming:

$$\min_{x \in \mathbb{R}^n} c^T x \text{ subject to } Ax = b,\ x \geq 0,$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

LP became important during and after WWII for solving practical problems of resource allocation, and has remained a central problem ever since.

The modern history of optimization dates to the mid-late 1940s, with the formulation of LP and discovery of the simplex method (Dantzig).

Some History: Theory and Practice in Optimization

Optimization formulations arose from the needs of applications - starting with LP. The formulations provide mathematical abstractions around which theory and algorithms can be developed.

Optimality conditions quickly follow formulations. e.g. theory for optimality of nonlinear programming (Karush-Kuhn-Tucker, 1948-51).

(Theory around formulations continues to be an important topic, e.g. weaker optimality conditions, sensitivity to data perturbations; well posedness, degeneracy, properties that promote faster convergence,...)

Iterative algorithms (starting with simplex for LP) are founded in optimality conditions. Information about the functions (and their gradients) is used to generate a sequence of points whose limits or accumulation points satisfy optimality conditions.

Much optimization research is about devising algorithms that are "efficient" for certain problem classes or paradigms; using math to analyze their behavior; and applying them to important practical problems.

What Kinds of Convergence Theory?

Theory for nonlinear optimization algorithms 1950s-1990s made standard assumptions (usually: C^2 functions) and derived two main kinds of results:

- asymptotic behavior of the iteration sequence: accumulation points / limits satisfy certain optimality conditions.
- local rates: linear, sublinear, superlinear convergence of sequences to their limits (where some additional properties are satisfied at the limit). Convergence to a nearby solution from a good initialization.

Also tighter analyses of important subclasses of the main problem classes.

These kinds of analysis shed light on some important features of algorithms. But they leave other important questions unanswered, e.g.

- How many iterations / how much computation will I need to find an approximate solution?
- Given the type of problem I am solving, which algorithms make the best use of the information I can provide about the functions?

Complexity in Nonlinear Optimization

Complexity results are a refinement of the asymptotic analysis, typically requiring some "global" assumptions on the properties of the functions.

The additional assumptions allow quantitative guarantees to be made about progress toward optimality at each iteration — not just the possibly minuscule improvement available for generic smooth functions.

For nonlinear optimization, these concepts arose in the USSR [Nemirovski and Yudin, 1983] and became popular in the west later, particularly after the machine learning community became heavily engaged in optimization (2009-)

(In linear programming, complexity was always a central issue.)

Additional Assumptions.

Typical additional assumptions:

- gradients are Lipschitz continuous ("L-smooth")

$$\|\nabla f(y) - \nabla f(z)\| \le L\|y - z\|;$$

- strong convexity:

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}\sigma\|x - y\|^2 \text{ for all } x, y \text{ and some } \sigma > 0.$$

Also higher-order smoothness results; results on the geometry of the feasible set (defined by the constraints); sharpness; PL and KL conditions; ....

Major Focus

We focus mainly on unconstrained optimization of a smooth function:

$$\min_x \; f(x), \quad \text{where } f : \mathbb{R}^n \to \mathbb{R} \text{ is smooth.}$$

There are **many** issues concerning the relationship between theory and practice of algorithms that arise even in this setting.

Interesting cases (ubiquitous in ML) where f has additional structure:

$$\text{expectation:} \quad f(x) = \mathbb{E}_\xi h(x;\xi),$$

$$\text{finite sum:} \quad f(x) = \frac{1}{N}\sum_{j=1}^{N} f_j(x).$$

(Approximate) Solutions: Unconstrained Optimization

Consider various conditions for optimality, and approximate optimality:

- Function suboptimality: $f(x) - f^* \le \epsilon$, where f^* is the optimal function value and $\epsilon > 0$.
- Distance to solution: $\text{dist}(x, \mathcal{S}) = \min_{x^* \in \mathcal{S}} \|x - x^*\| \le \epsilon$, where $\mathcal{S}$ is the set of minimizers of $f(x)$.
- First-order condition: $\nabla f(x) = 0$.
    - When f is convex, this condition is sufficient for x to be a minimizer of f.
    - Approximate first-order condition: $\|\nabla f(x)\| \le \epsilon_g$.
- Second-order condition: $\nabla^2 f(x) \succeq 0 \Leftrightarrow \lambda_{\min}(\nabla^2 f(x)) \ge 0$.
    - For smooth nonconvex functions, this is necessary for x to be a local minimizer.
    - Sufficient condition has strict inequality: $\lambda_{\min}(\nabla^2 f(x)) > 0$.
    - Approximate second-order condition: $\nabla^2 f(x) \succeq -\epsilon_H I$.

Why Approximate Solutions?

For nonlinear problems, iterative algorithms converge to a solution only asymptotically — generally, no iterate x^k actually satisfies optimality conditions exactly.

But for strictly positive tolerances ϵ (or ϵ_g or ϵ_H) these algorithms typically arrive at approximate optimality within a finite number of iterations — this number being a function of ϵ.

- Typically $O(\epsilon^{-1})$ or $O(\epsilon^{-2})$ or $O(|\log \epsilon|)$.

Convergence Rates and Complexity Bounds

Complexities can be given either in rates as a function of iteration k, or bound on number of iterations required for an ϵ-approximate solution.

Easy to translate between the two.

Let $\tau_k > 0$ is the quantity whose bound is decreasing to zero with k, e.g.

$$\tau_k = \|\nabla f(x^k)\|, \quad [f(x^k) - f^*], \quad \text{or} \ \|x^k - x^*\|.$$

When is $\tau_k \le \epsilon$?

sublinear:	$\tau_k \le \frac{A}{k}$	$: k \ge \frac{A}{\epsilon}$;
sublinear:	$\tau_k \le \frac{A}{\sqrt{k}}$	$: k \ge \frac{A^2}{\epsilon^2}$;
sublinear:	$\tau_k \le \frac{A}{k^2}$	$: k \ge \frac{\sqrt{A}}{\sqrt{\epsilon}}$;
linear:	$\tau_k \le (1-\phi)^k \tau_0$	$: k \ge \frac{1}{\phi} \lvert \log(\epsilon/\tau_0) \rvert$
quadratic:	$\tau_k \le \psi \tau_{k-1}^2$	$: k \ge \log \lvert \log \epsilon \rvert$.

Oracle Complexity for Nonlinear Optimization

Arithmetic measures of complexity are popular in such problems as linear programming, convex quadratic programming, linear complementarity.

But for nonlinear optimization, we cannot usually talk about arithmetic measures of complexity, because the arithmetic cost of evaluating function information varies greatly across problem instances.

[Nemirovski and Yudin, 1983] propose an oracle model, based on a well defined "unit of information" (oracle) about the function f, for example:

- evaluation of $f(x)$ at a given point x;
- evaluation of $(f(x), \nabla f(x))$ at a given x;
- evaluation of an unbiased estimate g of $\nabla f(x)$ at a given x.

Oracle complexity of an algorithm is a bound on the number of such units needed to identify an ϵ-approximate solution.

The concept of evaluation complexity is similar — the thing we are "evaluating" can be viewed as an oracle.

Iteration Complexity

Upper-bound the number of iterations of a given algorithm to find an ϵ-approximate solution.

- For nonlinear problems, algorithms often have two or three nested levels of iteration. The most useful iteration complexity analyses find bounds on the number of iterations at each level (or the total number of inner iterations).
- Iteration complexity are not so appealing when each iteration may require solution of a nontrivial problem e.g. finding the global minimizer of some high-degree polynomial.

For some f (particularly f arising in machine learning), the cost of linear algebra can be significant too (but usually doesn't change the overall complexity picture).

Gradient and Momentum Methods

To minimize f, each step of a gradient descent (GD) method has the form

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k).$$

Each step of a momentum method combines the latest gradient direction ∇f with the previous step which, in turn, encodes all previous gradients:

Heavy-ball, conjugate gradient:

$$x^{k+1} = x^k + \beta_k(x^k - x^{k-1}) - \alpha_k \nabla f(x^k).$$

Nesterov accelerated gradient:

$$x^{k+1} = x^k + \beta_k(x^k - x^{k-1}) - \alpha_k \nabla f(x^k + \beta_k(x^k - x^{k-1})).$$

(Choices of α_k and β_k are critical.)

Complexity: Upper and Lower Bounds

The term “complexity” usually refers to upper bounds on the work required by a particular algorithm on a given problem class.

But there is also interest in lower bounds, which are typically defined for an algorithm class / problem class pair.

Lower bounds usually derived from a worst-case instance:

> *“There is an instance in the given problem class for which all algorithms in the algorithm class require at least $N(\epsilon)$ oracle calls to find an ϵ-approximate solution.”*

An optimal algorithm is one in the algorithm class for which the upper and lower bounds are the same (within a constant) on the problem class. Nesterov’s accelerated gradient [Nesterov, 1983] is optimal for

- Problem class: Minimization of 1-smooth convex functions f;
- Algorithm class: Methods for which $x^k - x^0$ lies in the span of all gradients encountered up to this iteration.

Lower Bound Instance for First-Order Gradient Methods

$f(x) = \frac{1}{2}x^T Ax - e_1^T x$ (so $\nabla f(x) = Ax - e_1$), where

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & \dots & & & -1 & 2 & -1 \\ 0 & \dots & & & 0 & -1 & 2 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Solution x^* has components $x^*_{n-i+1} = \frac{i}{n+1}$, for $i = 1, 2, \dots, n$.

For $x^0 = 0$, kth iterate of a first-order method can have nonzero entries only in its first k components, so

$$\|x^k - x^*\|^2 \geq \sum_{j=k+1}^{n} (x_j^*)^2 \geq \frac{1}{8}\|x^0 - x^*\|^2,$$

$$f(x^k) - f^* \geq \frac{3}{8(k+1)^2}\|x^0 - x^*\|^2, \quad k = 1, 2, \dots, \frac{n}{2} - 1.$$

Lower-Bound Instance

The $1/k^2$ rate for $f(x^k) - f^*$ matches the upper bound.

Not fully satisfying because it only "works" over the first $n/2$ iterations, which is not always an interesting regime.

Moreover, if we make small random perturbations to the gradients, they "fill in" and the argument based on sparsity fails.

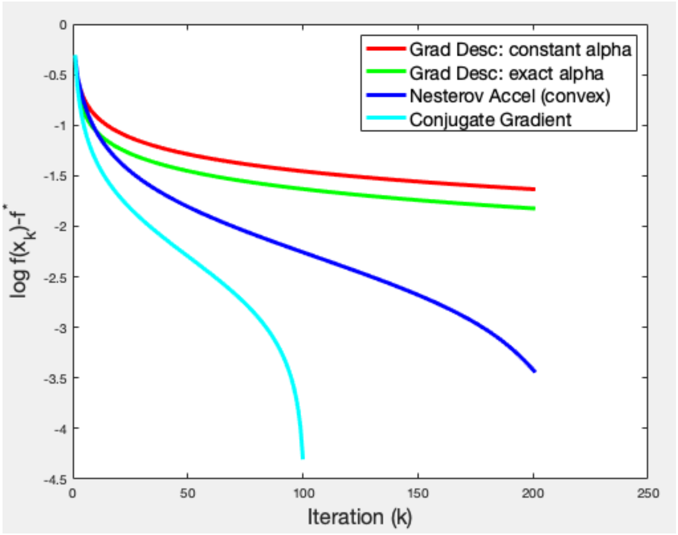
- Can be fixed (e.g. [Woodworth, 2021]) by generalizing to an argument based on subspaces.

Finding the optimal linear combination of gradients sounds unrealistic....

- But conjugate gradient actually achieves property this when f is convex quadratic!

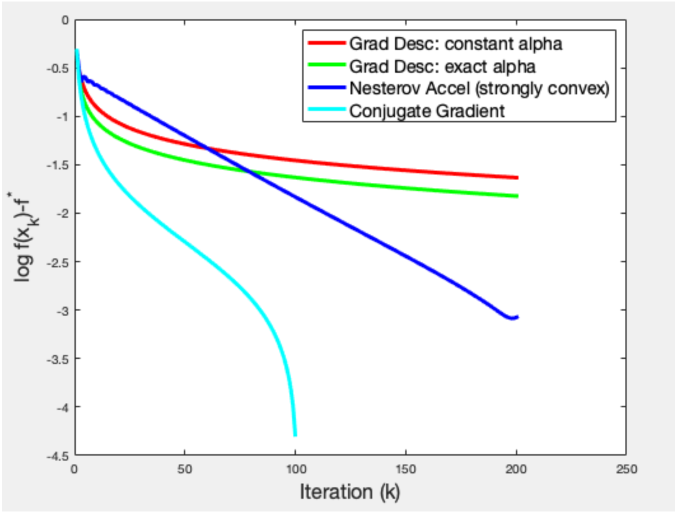
Results for the Worst-Case Example ($n = 100$)

Nesterov's method for weakly convex functions does better than GD (at least after early iterations).



Results for the Worst-Case Example ($n = 100$)

Nesterov's method for strongly convex is *worse* over the first $n/2$ iterations, though better than GD asymptotically.



Upper Bounds: Loose or Tight?

Sometimes the upper bounds reflect typical practical behavior.

In other cases the bounds are loose — practical behavior of an algorithm is much better than the complexity analysis suggests. WHY?

- Conservative assumptions about function properties e.g. L-smoothness. Local geometry may allow much more progress to be made at many iterations than worst-case analysis suggests.
- Adaptive algorithms can exploit variation of function properties across the domain. These gains are reflected in practice but it is hard to find an abstraction that allows theoretical analysis.
  - Theory by itself may not lead to good algorithm design.
- The worst-case instances are rare in the function class.
  - Simplex method for linear programming is the classic example of bad but rare worst cases.
- The problem has some additional (hidden?) structure that can be exploited to get tighter complexity bounds.

Complexity in Optimization: pre-2008

In LP, theoretical CS researchers were always keenly interested in complexity, and the properties of the simplex method.

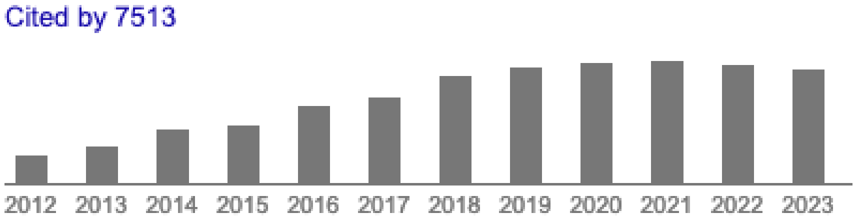
In optimization, much work on complexity for convex problems:

- The polynomial-time revolution in LP [Khachiyan, 1979], [Karmarkar, 1984] sparked a lot of work in the optimization community, on developing new LP methods and extending to semidefinite programming and conic optimization.
    - Including researchers with a “nonlinear” background.
- [Nemirovski and Yudin, 1983] was the founding document of “oracle complexity” (see above).
- The interior-point book of [Nesterov and Nemirovskii, 1994] was extremely influential in promoting a complexity perspective for convex nonlinear problems (including constrained problems). This emphasis was reflected in [Boyd and Vandenberghe, 2003].

Nesterov's 2004 Book

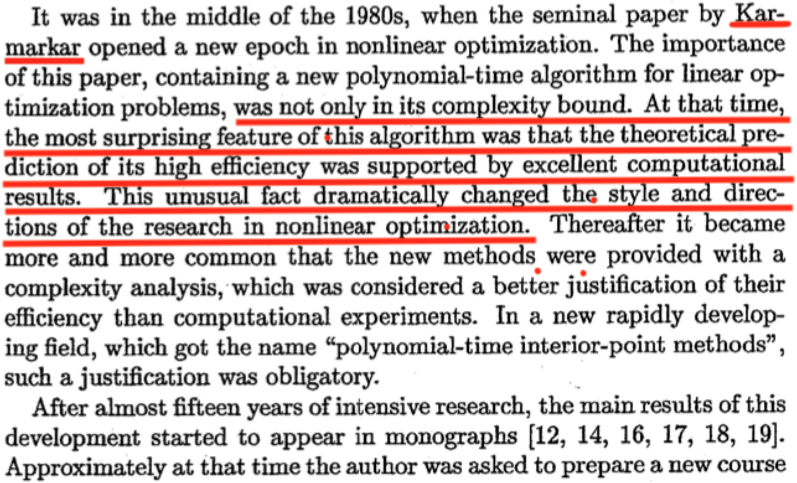
[Nesterov, 2004] deals largely with complexity issues for nonlinear convex optimization, with structured nonsmoothness.

Citations show the growth of interest in this topic!



In the foreword, Nesterov credits Karmarkar's paper — particularly its conjunction of good complexity and good computational results — with the growth of interest in complexity analysis.

(Prescient as usual - interest grew much more after 2004!)



It was in the middle of the 1980s, when the seminal paper by Karmarkar opened a new epoch in nonlinear optimization. The importance of this paper, containing a new polynomial-time algorithm for linear optimization problems, was not only in its complexity bound. At that time, the most surprising feature of this algorithm was that the theoretical prediction of its high efficiency was supported by excellent computational results. This unusual fact dramatically changed the style and directions of the research in nonlinear optimization. Thereafter it became more and more common that the new methods were provided with a complexity analysis, which was considered a better justification of their efficiency than computational experiments. In a new rapidly developing field, which got the name "polynomial-time interior-point methods", such a justification was obligatory.

After almost fifteen years of intensive research, the main results of this development started to appear in monographs [12, 14, 16, 17, 18, 19]. Approximately at that time the author was asked to prepare a new course

What Changed Since 2008?

- Interaction between machine learning and optimization (OptML) expanded greatly. Ongoing!
  - Joint sessions at 2009 ISMP in Chicago (organized by Bennett, Scheinberg, Wright).
  - [Nemirovski et al., 2009] — a study of stochastic gradient (SGD) appeared in SIOPT, at a time when SGD was emerging as **the** essential algorithm in ML.
  - Optimization saw an influx of researchers from ML and theoretical CS, who brought an intense interest in complexity.
- Growing interest in complexity for nonconvex optimization.
  - [Nesterov and Polyak, 2006] was highly influential.
  - Ubiquity of benign nonconvexity.

OptML: Stochastic Gradient (SGD)

A quest to improve complexity bounds led directly to new optimization algorithms for convex finite-sum problems that arise in ML:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

(where x are model parameters, each f_i depends on a single item of data.)

The basic SGD algorithm uses $\nabla f_{i_k}(x^k)$ for some randomly chosen $i_k \in \{1, 2, \dots, n\}$ as an unbiased proxy for $\nabla f(x^k)$. Step is

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k).$$

Later methods combine full-gradient descent (GD) with SGD. Better complexity + better numerical performance in some settings.

- SAG [Schmidt et al., 2016]
- SVRG [Johnson and Zhang, 2013]

These and other extensions are used for nonconvex problems too (neural networks), but the analysis was not extended until later, by Lan et al.

OptML: Coordinate Descent

Coordinate descent (CD) for $\min_{x \in \mathbb{R}^n} f(x)$ was popular in applications for many years. Iteration k:

- choose index $i_k \in \{1, 2, \dots, n\}$, randomly or cyclically;
- step $x^{k+1} = x^k - \alpha_k \nabla_{i_k} f(x^k) e_{i_k}$ where e_{i_k} is vector of all 0 except for 1 in position i_k. *(Just one coordinate changes at each iteration.)*

Little convergence analysis until until [Nesterov, 2012] (randomized), [Beck and Tetruashvili, 2013] (cyclic). Review paper: [Wright, 2015].

In important ML applications, CD has potential complexity advantages over full-gradient descent (GD). Better CD variants over the past 10 years have been motivated by both:

- improving worst-case complexity;
- stronger computational performance.

There has been a synergy between the theory and practice.

OptML: Coordinate Descent

Convergence rates of different variants of CD together with empirical studies for [Wright, 2015] yielded some deeper insights.

- **Cyclic (C):** Cycle through indices: $1, 2, 3, \dots, n, 1, 2, 3, \dots$;
- **Random (R):** Choose index i_k randomly with replacement from $\{1, 2, \dots, n\}$.
- **Random Permutations (RP):** Within each epoch of n iterations, choose index i_k randomly without replacement from $\{1, 2, \dots, n\}$.

Convergence theory for **C** can be applied trivially to **RP** too.

On convex quadratic f, theoretical rates are slower for **C** / **RP** than for **R**. But most problems show little difference in convergence behavior between the three variants.

Exception: When the Hessian has one dominant eigenvalue whose eigenvector is in general orientation, **C** achieves its worst-case (slower) behavior. But rates for **RP** track rates for **R**!

Explained in [Lee and Wright, 2018], [Wright and Lee, 2020].

OptML: Parallel Algorithms

Since 2011, there has been a huge literature on parallel variants of SGD and CD, in which complexity analysis plays a central role.

Motivation: Problems in ML are huge and need parallelism in practice.

- Anticipated in [Bertsekas and Tsitsiklis, 1989]
    - (not much about complexity there)
- Parallel Asynchronous SGD: HOGWILD! [Niu et al., 2011].
- Parallel Asynchronous CD: [Liu et al., 2015, Liu and Wright, 2015].
- Many parallel synchronous variants too.

Parallel SGD techniques central to federated learning.

Computation and communication considerations in FL are complicated, and many complex variants of SGD have been proposed that suit different platforms and contexts. [Woodworth, 2021] has some lower-bound theory.

Practice tends to drive theory in this area!

OptML: Benign Nonconvexity

Not much interesting complexity in finding the the global minimizer of general nonconvex functions [Nemirovski and Yudin, 1983] — it's hard!

But many nonconvex problems in ML are benign, easily solved despite the nonconvexity.

- matrix problems with explicit low-rank ("Burer-Monteiro") parametrizations e.g. [Burer and Monteiro, 2003], [Burer and Monteiro, 2005], [Chi et al., 2019];
- tensor problems with certain low-rank decompositions e.g. [Han et al., 2020];
- phase retrieval [Candès et al., 2015]; phase synchronization [Boumal, 2016];
- AC power flow e.g. [Zhou and Low, 2021];
- dictionary learning e.g. [Sun et al., 2016];
- neural networks (NNs).

Ju Sun's excellent page: `https://sunju.org/research/nonconvex/`

Benign Nonconvexity: Structures and Properties

For these and other nonconvex problems, there are algorithms that can find useful solutions with interesting complexity bounds.

Several structures and properties promote benign nonconvexity:

- All local minima are global minima in some problems.
- All saddle points are *strict* saddle points, so are easy to escape from (e.g. by detecting negative curvature in the Hessian, or making a small random step);
- Polyak-Łojasiewicz condition: $\|\nabla f(x)\|^2 \geq 2c(f(x) - f^*)$.
- Correlated gradient (regularity) condition:

$$\nabla f(x)^T (x - x^*) \geq \alpha \|x - x^*\|_2^2 + \beta \|\nabla f(x)\|_2^2, \quad \text{some } \alpha, \beta > 0;$$

- Some problems allow smart initialization schemes that place x^0 near a global minimizer.

Here, empirical observations are giving rise to interesting new theory!

Example: Matrix Sensing

Recall Matrix Sensing (Application III):

$$\min_X \frac{1}{2m} \sum_{j=1}^{m} (\mathcal{A}_j(X) - y_j)^2.$$

- When symmetric X has low rank r, write $X = ZZ^T$ where $Z \in \mathbb{R}^{n \times r}$.
- $\mathcal{A}_j(X) = \langle A_j, ZZ^T \rangle$ for some symmetric $A_j \in \mathbb{R}^{n \times n}$.
- Assume that the A_j satisfy a restricted isometry property (RIP):

$$(1 - \delta_q)\|X\|_F^2 \le \frac{1}{m} \sum_{j=1}^{m} \langle A_j, X \rangle^2 \le (1 + \delta_q)\|X\|_F^2,$$

  for all X with rank at most q and some $\delta_q \in (0, 1)$.

Formulation is thus

$$\min_Z h(Z) := \frac{1}{2m} \sum_{j=1}^{m} (\langle A_j, ZZ^T \rangle - y_j)^2.$$

Example: Matrix Sensing

If the properties above hold with $q = 2r$ and $\delta_{2r} \in (0, .1]$, then

- All local minima of F are global;
- All stationary points of F that are not strict have negative curvature in $\nabla^2 F(Z)$.

[Bhojanapalli et al., 2016]

Can initialize cleverly: The matrix

$$Y := \frac{1}{m} \sum_{j=1}^{m} y_j A_j$$

is close to the solution ZZ^T if the conditions above hold. Steepest descent on F converges from such a starting point.

Benign Nonconvexity in Overparametrized Neural Networks

- Modern neural networks have many more parameters (weights) x than needed to exactly fit the training data.
- The finite-sum loss function $N^{-1} \sum_{i=1}^{N} f_i(x)$ to be minimized is nonconvex.
- Still, gradient methods and SGD, if run long enough, achieve zero loss. That is, they fit the training data perfectly!
- Shockingly, they don't overfit. The solutions found by these methods still do well at predicting data outside the training set (but from the same distribution). Implicit regularization.

Understanding these phenomena in theoretical deep learning is engaging the interest of some top people in statistics, learning, PDEs, ... Examples: [Belkin et al., 2018], [Belkin et al., 2019],[Belkin, 2021], [Bartlett et al., 2021] [Chizat and Bach, 2020], [Mei et al., 2018], [Weinan et al., 2020a], [Weinan et al., 2020b].

Complexity for Smooth Nonconvex Optimization

Motivated by [Nesterov and Polyak, 2006], there are many works on complexity bounds for methods that find approximate second-order points:

$$\|\nabla f(x)\| \leq \epsilon_g, \quad \nabla^2 f(x) \succeq -\epsilon_H I.$$

These methods vary in practicality. Some are motivated by a desire to optimize complexity; others are derived from known practical approaches.

[Nesterov and Polyak, 2006] finds step p by solving the quadratic Taylor-series expansion added to a cubic term:

$$p^k = \arg\min_p \; \nabla f(x^k)^T p + \frac{1}{2} p^T \nabla^2 f(x^k) p + \frac{1}{6} M \|p\|^3,$$

where M is a Lipschitz constant for $\nabla^2 f$.

When $\epsilon_H = \epsilon_g^{1/2}$, requires $O(\epsilon_g^{-3/2})$ iterations.

Actually proposed much earlier [Griewank, 1981], with different motivation.

Smooth Nonconvex: Other Approaches

- Line-search with negative curvature [Royer and Wright, 2018, Royer et al., 2020];
- Trust region [Agarwal et al., 2016], [Curtis et al., 2021];
- Gradient descent with perturbations to avoid saddle points [Jin et al., 2017a, Jin et al., 2017b]
- Local convexification [Carmon et al., 2018].

Typically have iteration complexity $O(\epsilon_g^{-3/2})$ or $O(\epsilon_g^{-2})$.

Some have oracle complexity bounds too (where the "oracle" is a gradient evaluation or a Hessian-vector product). This is typically $O(\epsilon_g^{-7/4})$.

These bounds are highly pessimistic, but can be improved in certain settings, including some benign nonconvex settings, e.g. [O'Neill and Wright, 2023] obtains $O(|\log \epsilon|)$ bounds for low-rank matrix problems.

Pessimistic (or Nonexistent) Complexity Bounds Arise Elsewhere in Nonconvex Optimization

Newton and quasi-Newton methods are highly successful on nonconvex optimization, but convergence theory is only local.

- except that when a self-concordance property is satisfied, can prove global complexity for Newton's method.

Limited-memory quasi-Newton (L-BFGS) doesn't even have interesting local convergence theory: Essentially, "*with safeguards, L-BFGS is not too much worse than gradient descent*." In practice it is much better.

Convergence / complexity results for nonlinear conjugate gradient methods are also limited, but these methods often work well on large problems.

- [Nemirovski and Yudin, 1983] give a lower-bound case in which it is no better than gradient descent.
- [Karimi and Vavasis, 2021] for convex problems: Nonlinear CG with temporary switching to accelerated gradient when insufficient progress is made. Gets optimal rates for convex and strongly convex, plus good practical performance.

Pursuit of Good Complexity Doesn't Necessarily Lead to Good Algorithms

The quest for methods that achieve optimal theoretical complexity often does not yield good practical methods — though the theory can be interesting, and can lead to practical methods later.

- LP Ellipsoid Method: [Khachiyan, 1979]
- Sorting Networks - networks that sort n numbers by a succession of pairwise comparisons.
  - Simple networks (e.g. bitonic) do $O(n \log^2 n)$ comparisons.
  - Optimal networks (AKS) require $O(n \log n)$, but there is a constant 10^{60} in the $O()$ term.
- Log-barrier for bound-constrained minimization [O'Neill and Wright, 2021] optimal in theory, but much slower than the projected Newton method in [Xie and Wright, 2021].

Closing the Gaps Between Theory and Practice

There is a lot of interesting research still to do in closing gaps between complexity upper bounds and practical behavior of algorithms.

- Are the hard problems rare? ("Small measure" in the space of problem data?)
- Can all hard problems be converted to easy problems via a small perturbation? (Smoothed analysis)
- Can we close the gap by refining / dividing the problem class?
- Can we explain observed differences in algorithm behavior by theory?

Examples of the last point:

- Weird behavior of coordinate-descent variants on convex quadratics with one dominant eigenvalue: eventually explained in [Lee and Wright, 2018], [Wright and Lee, 2020].
- Powell's work on the difference between BFGS and DFP quasi-Newton methods in the mid-1980s.

Conclusions

Fletcher (1987): Optimization is a "fascinating blend of theory and computation, heuristics and rigor."

(Quoted in the foreword of [Nocedal and Wright, 1999].)

Interplay between theory and practice has always driven optimization.

Within optimization — particularly optimization problems from ML, and nonconvex problems — complexity theory has attained prominence as a means to design interesting algorithms and understand their properties.

"Engineering" algorithms is still important to their practical applicability.

Much work remains in using complexity and other theoretical perspectives to explain the behavior of interesting algorithms on interesting problems.

References I

Agarwal, A., Allen-Zhu, Z., Bullins, B., Hazan, E., and Ma, T. (2016).
Finding local minima for non-convex optimization in linear time.
Preprint arXiv:1611.01146, Princeton University.

Bartlett, P. L., Montanari, A., and Rakhlin, A. (2021).
Deep learning: a statistical viewpoint.
arXiv preprint arXiv:2103.09177.

Beck, A. and Tetruashvili, L. (2013).
On the convergence of block coordinate descent type methods.
SIAM Journal on Optimization, 23(4):2037–2060.

Belkin, M. (2021).
Fit without fear: The remarkable mathematical phenomenon of deep learning through the prism of interpolation.
Acta Numerica, 30(arXiv:2105.14368):203–248.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2018).
Reconciling modern machine learning and the bias-variance trade-off.
Technical Report arXiv:1812.11118, The Ohio State University.

Belkin, M., Hsu, D., and Xu, J. (2019).
Two models of double descent for weak features.
Technical Report arXiv:1903.07571, The Ohio State University.

References II

Bertsekas, D. P. and Tsitsiklis, J. N. (1989).
Parallel and Distributed Computation: Numerical Methods.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016).
Global optimality of local search for low-rank matrix recovery.
Technical Report arXiv:1605.07221, Toyota Technological Institute.

Blum, L. A., Cucker, F., Shub, M., and Smale, S. (1998).
Complexity and Real Computation.
Springer Science and Business Media.

Borgwardt, K.-H. (1987).
The Simplex Method, A Probabilistic Analysis.
Springer-Verlag, Berlin.

Boumal, N. (2016).
Nonconvex phase synchronization.
SIAM Journal on Optimization, 26(4):2355–2377.

Boyd, S. and Vandenberghe, L. (2003).
Convex Optimization.
Cambridge University Press.

References III

Burer, S. and Monteiro, R. D. C. (2003).
A nonlinear programming algorithm for solving semidefinite programs via low-rank factorizations.
Mathematical Programming, Series B, 95:329–257.

Burer, S. and Monteiro, R. D. C. (2005).
Local minima and convergence in low-rank semidefinite programming.
Mathematical Programming, Series A, 103:427–444.

Candès, E., Li, X., and Soltanolkotabi, M. (2015).
Phase retrieval via a Wirtinger flow.
IEEE Transactions on Information Theory, 61:1985–2007.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2018).
Accelerated methods for non-convex optimization.
SIAM Journal on Optimization, 28:1751–1772.

Chi, Y., Lu, Y. M., and Chen, Y. (2019).
Nonconvex optimization meets low-rank matrix factorization: An overview.
IEEE Transactions in Signal Processing, 67:5239–5269.

Chizat, L. and Bach, F. (2020).
Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss.
In *Conference on Learning Theory*, pages 1305–1338. PMLR.

References IV

Cohen, M. B., Lee, Y. T., and Song, Z. (2021).
Solving linear programs in the current matrix multiplication time.
Journal of the ACM (JACM), 68(1):1–39.

Curtis, F. E., Robinson, D. P., Royer, C. W., and Wright, S. J. (2021).
Trust-region Newton-CG with strong second-order complexity guarantees for nonconvex optimization.
SIAM Journal on Optimization, 31:518–544.

Frisch, K. R. (1955).
The logarithmic potential method of convex programming.
Technical Report, University Institute of Economics, Oslo, Norway.

Griewank, A. (1981).
The modification of Newton's method for unconstrained optimization by bounding cubic terms.
Technical Report NA/12, DAMTP, Cambridge University.

Han, R., Willett, R., and Zhang, A. (2020).
An optimal statistical and computational framework for generalized tensor estimation.
arXiv preprint arXiv:2002.11255.

References V

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and I., J. M. (2017a).
How to escape saddle points efficiently.
In *Proeedings of the 34th International Conference on Machine Learning*, volume 70, pages 1724–1732.

Jin, C., Netrapalli, P., and Jordan, M. I. (2017b).
Accelerated gradient descent escapes saddle points faster than gradient descent.
Technical Report arXiv:1711.10456, UC-Berkeley.

Johnson, R. and Zhang, T. (2013).
Accelerating stochastic gradient descent using predictive variance reduction.
In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc.

Karimi, S. and Vavasis, S. A. (2021).
Nonlinear conjugate gradient for smooth convex functions.
Techincal Report arXiv 2111.11613, University of Waterloo.

Karmarkar, N. (1984).
A new polynomial-time algorithm for linear programming.
Combinatorica, 4:373–395.

References VI

Khachiyan, L. G. (1979).
A polynomial algorithm in linear programming.
Soviet Mathematics Doklady, 20:191–194.

Klee, V. and Minty, G. J. (1972).
How good is the simplex algorithm?
In Shisha, O., editor, *Inequalities*, pages 159–175. Academic Press, New York.

Lee, C.-p. and Wright, S. J. (2018).
Random permutations fix a worst case for cyclic coordinate descent.
IMA Journal of Numerical Analysis, 39:1246–1275.

Liu, J. and Wright, S. J. (2015).
Asynchronous stochastic coordinate descent: Parallelism and convergence properties.
SIAM Journal on Optimization, 25(1):351–376.

Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. (2015).
An asynchronous parallel stochastic coordinate descent algorithm.
Journal of Machine Learning Research, 16:285–322.
arXiv:1311.1873.

Mei, S., Montanari, A., and Nguyen, P.-M. (2018).
A mean field view of the landscape of two-layer neural networks.
Proceedings of the National Academy of Sciences, 115(33):E7665–E7671.

References VII

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009).
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on Optimization, 19(4):1574–1609.

Nemirovski, A. S. and Yudin, D. B. (1983).
Problem Complexity and Method Efficiency in Optimization.
John Wiley.

Nesterov, Y. (1983).
A method for unconstrained convex problem with the rate of convergence $O(1/k^2)$.
Doklady AN SSSR, 269:543–547.

Nesterov, Y. (2004).
Introductory Lectures on Convex Optimization: A Basic Course.
Springer Science and Business Media, New York.

Nesterov, Y. (2012).
Efficiency of coordinate descent methods on huge-scale optimization problems.
SIAM Journal on Optimization, 22:341–362.

Nesterov, Y. and Nemirovskii, A. S. (1994).
Interior Point Polynomial Methods in Convex Programming.
SIAM Publications, Philadelphia.

References VIII

Nesterov, Y. and Polyak, B. T. (2006).
Cubic regularization of Newton method and its global performance.
Mathematical Programming, Series A, 108:177–205.

Niu, F., Recht, B., Ré, C., and Wright, S. J. (2011).
HOGWILD!*: A lock-free approach to parallelizing stochastic gradient descent*, pages 693–701.
Curran Associates.

Nocedal, J. and Wright, S. J. (2006).
Numerical Optimization.
Springer, New York, second edition.

O'Neill, M. and Wright, S. J. (2021).
A log-barrier Newton-CG method for bound constrained optimization with complexity guarantees.
IMA Journal of Numerical Analysis, 41:84–121.

O'Neill, M. and Wright, S. J. (2023).
A line-search descent algorithm for strict saddle functions with complexity guarantees.
Journal of Machine Learning Research, 24:1–34.
arXiv:2006.07925.

References IX

Royer, C. W., O'Neill, M., and Wright, S. J. (2020).
A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization.
Mathematical Programming, Series A, 180(arXiv:1803.02924):451–488.

Royer, C. W. and Wright, S. J. (2018).
Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization.
SIAM Journal on Optimization, 28:1448–1477.

Schmidt, M., Le Roux, N., and Bach, F. (2016).
Minimizing finite sums with the stochastic average gradient.
Mathematical Programming, 162:83–112.

Spielman, D. A. and Teng, S.-H. (2004).
Smoothed analysis of algorithms: Why the simplex method usually takes polynomial time.
Journal of the Association for Computing Machinery, 51(3):385–463.

Sun, J., Qu, Q., and Wright, J. (2016).
Complete dictionary recovery over the sphere i: Overview and the geometric picture.
IEEE Transactions on Information Theory, 63(2):853–884.

References X

van den Brand, J. (2020).
A deterministic linear program solver in current matrix multiplication time.
In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM.

van den Brand, J. (2021).
Unifying matrix data structures: Simplifying and speeding up iterative algorithms.
In *Symposium on Simplicity in Algorithms (SOSA)*, pages 1–13. SIAM.

van den Brand, J., Lee, Y. T., Sidford, A., and Song, Z. (2020).
Solving tall dense linear programs in nearly linear time.
In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 775–788.

Weinan, E., Ma, C., Wojtowytsch, S., and Wu, L. (2020a).
Towards a mathematical understanding of neural network-based machine learning: What we know and what we don't.
arXiv preprint arXiv:2009.10713.

Weinan, E., Ma, C., and Wu, L. (2020b).
A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics.
Science China Mathematics, pages 1–24.

References XI

Woodworth, B. (2021).
The Minimax Complexity of Distributed Optimization.
PhD thesis, Toyota Technological Institute-Chicago.

Wright, S. J. (1996).
A path-following interior-point algorithm for linear and quadratic optimization problems.
Annals of Operations Research, 62:103–130.

Wright, S. J. (1997).
Primal-Dual Interior-Point Methods.
SIAM, Philadelphia, PA.

Wright, S. J. (2015).
Coordinate descent algorithms.
Mathematical Programming, Series B, 151:3–34.

Wright, S. J. and Lee, C.-p. (2020).
Analyzing random permutations for cyclic coordinate descent.
Mathematics of Computation, 89:2217–2248.

Xie, Y. and Wright, S. J. (2021).
Complexity of projected Newton methods in bound-constrained optimization.
Technical Report arXiv:2103:15989, University of Wisconsin-Madison.
In preparation.

References XII

Zhou, F. and Low, S. H. (2021).
Conditions for exact convex relaxation and no spurious local optima.
arXiv preprint arXiv:2102.11946.

Linear Programming: Optimality

$$\min_{x \in \mathbb{R}^n} c^T x \text{ subject to } Ax = b, \; x \geq 0.$$

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

Optimality conditions: $\exists \lambda \in \mathbb{R}^m$, $s \in \mathbb{R}^n$ such that

$$Ax = b, \tag{1a}$$

$$A^T \lambda + s = c, \tag{1b}$$

$$x_i s_i = 0, \quad i = 1, 2, \ldots, n, \tag{1c}$$

$$(x, s) \geq 0. \tag{1d}$$

System of (mildly) nonlinear equations, with bounds.

Approximate solutions to the LP typically satisfy all conditions except (1c), which is replaced by

$$\mu := \frac{1}{n} x^T s = \frac{1}{n} \sum_{i=1}^{n} x_i s_i \leq \epsilon, \quad \text{for some } \epsilon > 0.$$

Real-number model: LP

The real number complexity model of [Blum et al., 1998], applied to LP, assumes that

- Each (nonzero) entry in (A, b, c) requires 1 unit of storage.
- each arithmetic operation with one or two real numbers requires 1 unit of computation.

This model is much closer to floating-point computations than the traditional bit-complexity model, which assumes that the data (A, b, c) are rational and is motivated by the Turing machine model of computation.

For interior-point methods for LP, take product of the bound on the number of iterations and the linear algebra cost per iteration, which is (naively) $O(n^3)$.

Classically, get bounds between $O(n^{3.5}|\log \epsilon|)$ and $O(n^5|\log \epsilon|)$ real-number operations. (But see later!)

Simplex Method for LP: Complexity

The simplex method developed by Dantzig in the 1940s was (and is) highly effective in practice, typically requiring a modest multiple of n iterations. Despite much research over 75 years, no polynomial variant is known.

[Klee and Minty, 1972] devised an LP for which the number of simplex pivots is exponential in n.

Remarkable example of a wide gap between complexity theory and practical performance. Efforts to close it include:

- [Borgwardt, 1987] average-case analysis: average number of pivots over some distribution of LP data (A, b, c).
- [Spielman and Teng, 2004] smoothed analysis: "maximum over inputs of the expected performance of an algorithm under small random perturbations of that input." Show that simplex has smoothed complexity polynomial in n.
  (Also polynomial in 1/(std dev of the perturbations).)

LP in Polynomial Time

The poor worst-case complexity of simplex was a strong motivator for the search for polynomial-time methods.

The first — the ellipsoid method of [Khachiyan, 1979] — is impractical.

The "projective algorithm" of [Karmarkar, 1984] was a breakthrough.

- Claims for good practical performance too, not fully vindicated.
- But it connected to an earlier stream of work in optimization (log barrier: [Frisch, 1955]) and sparked development of related methods that quickly proved to be competitive: primal-dual interior-point (PDIP) methods.

PDIP methods apply a continuation technique to the KKT conditions (1), replacing complementarity condition $x_i s_i = 0$ by $x_i s_i = \mu > 0$ and

- Driving $\mu \downarrow 0$ in a carefully controlled way;
- Maintaining x and s strictly positive throughout: "interior-point";
- Taking Newton-like steps, with backtracking to keep x, s positive.

Complexity of PDIP Variants: Classical Approaches

Classical theory for PDIP (1988-1994) has iteration complexity ranging from $O(n^{1/2}|\log \epsilon|)$ to $O(n^2|\log \epsilon|)$, for different variants.

- There is an almost inverse relationship between "power of n" and "number of iterations required in practice."
- For a given method, number of iterations is almost independent of n.
- The analysis is elementary: an end-to-end complexity result can be proved in a single lecture of a graduate course.

Complexity of PDIP Variants: Newer Approaches

PDIP methods with improved complexity have been developed recently by Yin Tat Lee, Sidford, Madry, Cohen, Song, van den Brand, and others (see e.g. [van den Brand et al., 2020], [van den Brand, 2020], [Cohen et al., 2021]). Tied to

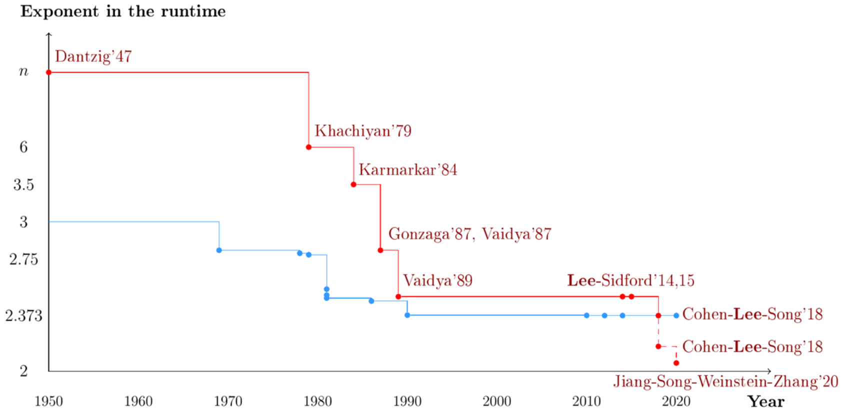
- cost of $n \times n$ matrix multiplication $O(n^{\omega})$ for $\omega \approx 2.38$, or
- number of nonzeros in A.

In fact, LP complexity has essentially caught up to matrix multiplication complexity (to within an $O(\log n)$ factor).

Recent Progress in LP Complexity (Yin Tat Lee, 2021)

min $c^T x$ s.t. $Ax = b$, $x \geq 0$.

Red: LP complexity; Blue: matrix multiplication complexity.



Current LP record of $2 + \frac{1}{18}$ assumes that matrix mult complexity is $\omega = 2$.

Theoretically Faster LP Methods

Some of these methods are based on practical PDIP methods, but

- Compute the iterates and Newton steps only approximately;
- Don't form and factor the coefficent matrices from scratch for the Newton steps — low-rank updates from earlier iterations give a sufficiently accurate step.
- Some use weighted variants of the central path;
- Some exploit special properties of particular problems e.g. max-flow.

Aaron Sidford: "An exercise in dynamic data structures" A unified description of factorization updating is [van den Brand, 2021].

Practice $\rightarrow$ Theory: What was thought of as a grungy numerical software implementation issue — maintaining matrix factorizations efficiently — is a key aspect of these theoretical developments.

The improvements are not (yet) relevant to practice.