Parameter Selection in Optimization

Stephen Wright

University of Wisconsin-Madison

NIPS Workshops, Dec 2016

Terminology

- $$f\left(\underset{\substack{\uparrow\\ \text{Variable}}}{x}^{k+1}\right) \leq f\left(x^{k}\right) + \underset{\substack{\uparrow\\ \text{Parameter}}}{c_1} \nabla f(x^{k})^{T} p^{k}$$

Parameters in Optimization Algorithms

- Almost all optimization algorithms contain parameters.
  - Adaptive parameter choices can be coded as heuristics, which are themselves parametrized.
- Often, convergence theory allows wide latitude in the choice of these parameters, but large variations in practical performance are seen over these parameter ranges.
- Sometimes, theory is just a (conservative) guide, and non-theoretical parameter choices are better.

We examine the role of parameters in a variety of continuous optimization algorithms, mostly deterministic.

- How sensitive is algorithm performance to parameters?
- How are good parameter values chosen in practice?
- What systematic efforts have been made to choose good parameters?
- Are there lessons for parameter choices in stochastic algorithms?

Outline

1. Measure the quality of parameters.
2. Parameters in several important algorithms
    - Primal-dual interior-point for LP
    - Accelerated gradient
    - Stochastic Gradient
    - Line-search methods for smooth unconstrained minimization.
    - Forward-backward Methods (SpaRSA) for ℓ_1 regularization.

“Good” and “Bad” Parameters

How do “good” and “bad” parameter choices differ?

- Efficiency: time / iterations required to solve a problem.
- Reliability: Does the algorithm crash, or does it solve the problem in a reasonable time?

Can conflate these two criteria:

- If the algorithm fails, could restart with different parameters.
- Can design meta-algorithms in which the main algorithm is run with different parameter settings (sequentially or concurrently), on where parameters are chosen adaptively.

How important are efficiency / reliability?

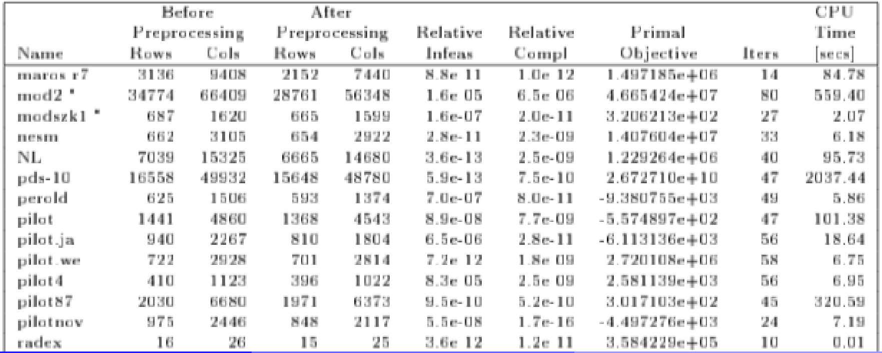
- Depends on the (expected) utility function that’s overlaid on the efficiency / reliability measures.
- Highly context-dependent. In some contexts, a factor-of-10 worse runtime makes little difference. In others, even a factor of 2 is bad.

Systematic Testing

Historically, comparison of optimization algorithms has been done using **batteries**: collections of "representative" problems.

(Could use the same techniques to compare different parameter choices within an algorithm.)

1980-2000: Usually tabulated test problems vs. iterations and runtimes on each problem. Here's part of a table from [Czyzyk et al., 1997]:



	Before Preprocessing		After Preprocessing		Relative	Relative	Primal		CPU Time
Name	Rows	Cols	Rows	Cols	Infeas	Compl	Objective	Iters	[secs]
maros r7	3136	9408	2152	7440	8.8e 11	1.0e 12	1.497185e+06	14	84.78
mod2 *	34774	66409	28761	56348	1.6e 05	6.5e 06	4.665424e+07	80	559.40
modszk1 *	687	1620	665	1599	1.6e-07	2.0e-11	3.206213e+02	27	2.07
nesm	662	3105	654	2922	2.8e-11	2.3e-09	1.407604e+07	33	6.18
NL	7039	15325	6665	14680	3.6e-13	2.5e-09	1.229264e+06	40	95.73
pds-10	16558	49932	15648	48780	5.9e-13	7.5e-10	2.672710e+10	47	2037.44
perold	625	1506	593	1374	7.0e-07	8.0e-11	-9.380755e+03	49	5.86
pilot	1441	4860	1368	4543	8.9e-08	7.7e-09	-5.574897e+02	47	101.38
pilot.ja	940	2267	810	1804	6.5e-06	2.8e-11	-6.113136e+03	56	18.64
pilot.we	722	2928	701	2814	7.2e 12	1.8e 09	2.720108e+06	58	6.75
pilot4	410	1123	396	1022	8.3e 05	2.5e 09	2.581139e+03	56	6.95
pilot87	2030	6680	1971	6373	9.5e-10	5.2e-10	3.017103e+02	45	320.59
pilotnov	975	2446	848	2117	5.5e-08	1.7e-16	-4.497276e+03	24	7.19
radex	16	26	15	25	3.6e 12	1.2e 11	3.584229e+05	10	0.01

Aggregating Performance Information

How to aggregate the performance on a battery of tests into more useful, "lower-dimensional" comparisons?

- Sum the runtimes. (Bad! Biased by performance on the long-running problems.)
- Rank solvers on each problem; aggregate the ranks e.g. average rank, count number of wins. (Still common).
- Performance profiles [Dolan and Moré, 2002]. Based on *relative* performance of different methods on each problem. Graphical rather than numerical. *Very popular!*

Performance Profiles [Dolan and Moré, 2002]

- Given S solvers on P problems, find the runtime

$$t_{p,s}, \quad p = 1, 2, \ldots, P, \;\; s = 1, 2, \ldots, S.$$

- Normalize the runtime for each problem relative to the best solver for that problem:

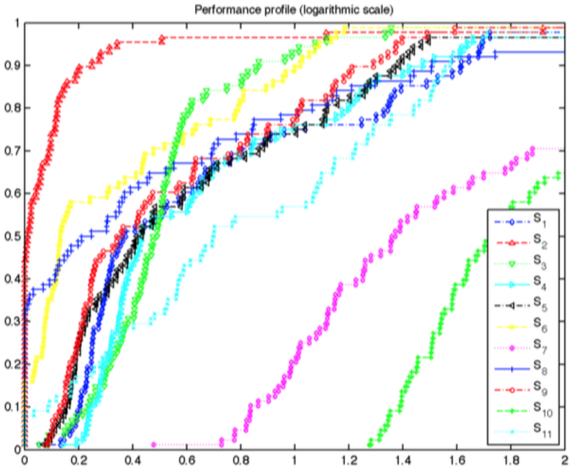
$$r_{p,s} := \frac{t_{p,s}}{\min_{j=1,2,\ldots,S} t_{p,j}}.$$

- For each solver $s = 1, 2, \ldots, S$, compute a cumulative distribution function:

$$\rho_s(\tau) := \frac{1}{P} \left| \{ p = 1, 2, \ldots, P \,:\, r_{p,s} \le \tau \} \right|.$$

- Graph $\rho_s(\tau)$ vs τ (or $\log_2 \tau$) for each $s = 1, 2, \ldots, S$.
- Instead of “runtime” could use other performance measures e.g. function evaluations, gradient evaluations.

Example from [Moré, 2007]



11 direct solvers for solvers for sparse linear systems. (X-axis is $\log_2 \tau$.)
(How to reduce this to a scalar metric?)

Are There Alternatives to Battery Testing?

In some applications of stochastic gradient, a single data set may suffice. (Could derive multiple instances from it.)

In other contexts, parametrized families of problems have been proposed. (More parameters!) See [Lyness and Kaganove, 1977] for numerical quadrature.

- These can be carefully controlled, so give insights into the workings of the algorithm....
- but may be too narrow in scope to give a broad indication of performance in the wild.

Case Study: Interior-Point for Linear Programming

Breakthroughs in practical primal-dual interior-point methods for LP happened in 1988-92. Continual refinement since then.

"Mehrotra predictor-corrector" (MPC) is a parametrized heuristic that, when properly tuned, gives reliable and fast local convergence.

$$\begin{aligned} \text{Primal:} \quad & \min_x c^T x \ \text{ s.t. } \ Ax = b, \ \ x \geq 0; \\ \text{Dual:} \quad & \max_{\lambda, s} b^T \lambda \ \text{ s.t. } \ A^T \lambda + s = c, \ \ s \geq 0, \end{aligned}$$

where A is $m \times n$, $x \in \mathbb{R}^n$, $s \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$.

Primal-dual optimality conditions for (x, λ, s):

$$Ax = b, \quad A^T \lambda + s = c, \quad (x, s) \geq 0, \quad XSe = 0,$$

where

$$X = \text{diag}(x_1, x_2, \ldots, x_n), \ \ S = \text{diag}(s_1, s_2, \ldots, s_n), \ \ e = (1, 1, \ldots, 1)^T.$$

Primal-Dual Interior-Point

At optimality, have $x_i s_i = 0$ for all $i = 1, 2, \ldots, n$. One of x_i, s_i is zero and the other is nonnegative. All iterates have (x, s) strictly positive. Use average of $x_i s_i$ as measure of optimality: $\mu = x^T s / n$: Duality Gap.

Ingredients for MPC: "Affine-scaling" step, which is a Newton direction for the equality optimality conditions:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x_{\text{aff}} \\ \Delta \lambda_{\text{aff}} \\ \Delta s_{\text{aff}} \end{bmatrix} = - \begin{bmatrix} Ax - b \\ A^T \lambda + s - c \\ -XSe \end{bmatrix}$$

Actual search direction $(\Delta x, \Delta \lambda, \Delta s)$ also has correction and centering:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} Ax - b \\ A^T \lambda + s - c \\ -XSe - \Delta X_{\text{aff}} \Delta S_{\text{aff}} e + \sigma \mu e, \end{bmatrix}$$

for some $\sigma \in (0, 1)$.

Key Parameters

Steps along the primal direction Δx and dual direction $(\Delta\lambda, \Delta s)$ need to maintain positivity of x and s components:

$$x + \alpha_{\text{pri}} \Delta x > 0, \quad s + \alpha_{\text{dual}} \Delta s > 0.$$

Simplest approach is to choose α_{pri} and α_{dual} to be the maximum value that satisfies these conditions scaled by a "backoff" factor γ slightly less than 1, e.g. $\gamma = .99$ or $\gamma = .999$.

Choice of σ is critical! [Mehrotra, 1992] discovered a very effective heuristic:

- Find maximum values of $\alpha_{\text{aff, pri}}$ and $\alpha_{\text{aff, dual}}$ such that

$$x + \alpha_{\text{aff, pri}} \Delta x_{\text{aff}} \geq 0, \quad s + \alpha_{\text{aff, dual}} \Delta s_{\text{aff}} \geq 0.$$

- Compute duality gap for this step:

$$\mu_{\text{aff}} := (x + \alpha_{\text{aff, pri}} \Delta x_{\text{aff}})^T (s + \alpha_{\text{aff, dual}} \Delta s_{\text{aff}})/n;$$

- Set $\sigma = (\mu_{\text{aff}}/\mu)^{\chi}$. (Exponent χ is a parameter. Default $\chi = 3$.)

[Wright, 1997, Chapter 10] partly describes state of the art around 1997.

Key Parameters

The PCx code [Czyzyk et al., 1997] has several other parameters, some hard-wired into the code:

- Multiple higher-order correction heuristic [Gondzio, 1996]
- Loop unrolling in the core linear algebra computations
- Parameters for sparse Cholesky factorization and iterative refinement. (Linear equations become ill-conditioned near the solution.)
- Parameters for handling dense columns in A.
- Tolerances in presolve, convergence tolerances.

Reliability is sensitive to these parameters! Sometimes there is failure to converge if e.g. the step scaling parameter γ is not close enough to 1. "Converge before the conditioning of the linear system gets too bad."

Efficiency is also sensitive to parameters, particularly those in the sparse linear algebra. Usually within a modest multiple.

Development of some codes 1988-98 "overfit" to the Netlib LP test set.

PDIP: Choosing These Parameters

- Many parameters!
- There are tricky interactions between them.
  - Example: choice of strategy in the linear system solve is *not* necessarily the same as what you would do when solving a symmetric positive definite linear system arising from an elliptic PDE. See e.g. [Wright, 1999].
- AFAIK, no systematic study of best choices for these parameters.[1] Experience of implementation experts over many years on client problems has likely resulted in effective choices.
- (Improvements in integer linear programming have been more dramatic.)

[1] But I've been out of the PDIP loop for a long time!

Accelerated Gradient

One variant of Nesterov's accelerated gradient for strongly convex, smooth f depends on parameters L and μ such that

$$\mu I \preceq \nabla^2 f(x) \preceq LI, \quad \text{for all } x.$$

Choose x^0, set $y^0 = x^0$, and iterate as follows:

$$x^{k+1} = y^k - \frac{1}{L}\nabla f(x^k), \quad y^{k+1} = x^{k+1} + \frac{\sqrt{L/\mu}-1}{\sqrt{L/\mu}+1}(x^{k+1} - x^k).$$

Convergence:

$$\begin{aligned} f(x^k) - f(x^*) &\le \frac{L+\mu}{2}\|x^0 - x^*\|^2 \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \\ &\le \frac{L+\mu}{\mu}(f(x^0) - f(x^*)) \left(1 - \sqrt{\frac{\mu}{L}}\right)^k. \end{aligned}$$

Accelerated Gradient

Consider convex strongly quadratic

$$f(x) = \frac{1}{2}x^T Ax - b^T x,$$

so that μ and L are the minimum and maximum eigenvalues of A.

On such functions Conjugate Gradient has convergence

$$\begin{aligned} f(x^k) - f(x^*) &\le 4\left(1 - \frac{2}{\sqrt{L/\mu}+1}\right)^{2k} (f(x^0) - f(x^*)) \\ &\approx 4\left(1 - 4\sqrt{\frac{\mu}{L}}\right)^{k} (f(x^0) - f(x^*)). \end{aligned}$$

Doesn't require estimates of μ and L. (Same complexity as Nesterov AG; rate is faster by a constant factor.)

Question: How is performance of Nesterov AG affected by the quality of estimates of L and μ?

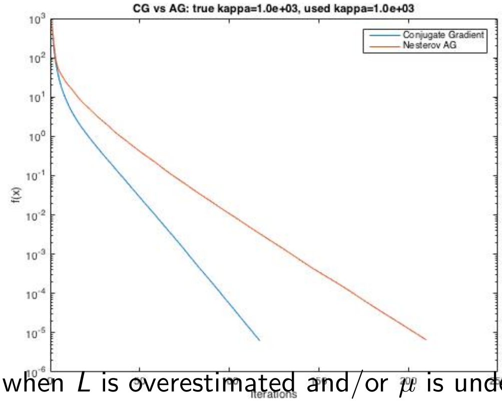
Tested this on a problem with $n = 10000$, $L = 1$, $\mu = 10^{-3}$.

Results

Declare convergence when $f(x^k) - f^* \leq 10^{-8}(f(x^0) - f^*)$.

Declare divergence when $f(x^k) - f^* \geq 10(f(x^0) - f^*)$.

CG: 115 iterations. AG with correct (L, μ) 207 iterations.



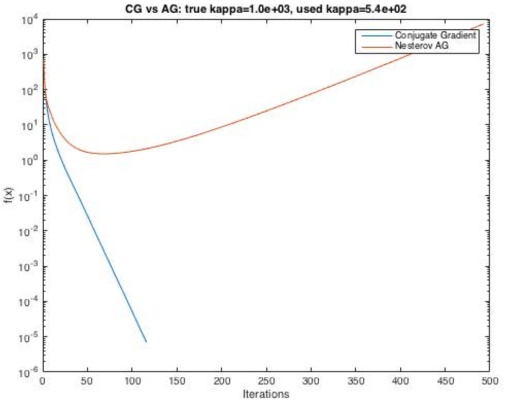
AG is still valid when L is overestimated and/or μ is undersestimated. Performance degrades gracefully in this regime.

Results: Invalid L or μ

Consider invalid settings: $L_{\text{fac}} > 1$, $\mu_{\text{fac}} > 0$ and

$$L = \lambda_{\max}(A)/L_{\text{fac}}, \quad \mu = \lambda_{\min}(A) * \mu_{\text{fac}}.$$

Example: $L_{\text{fac}} = \mu_{\text{fac}} = 1.36$, get divergence for AG:



Results

μ_{fac}	iters
1	207
1.5	278
2	345
3	442
10	828

(a) AG iterations: $L_{\text{fac}} = 1$

L_{fac}	iters
1.1	197
1.3	180
1.34	203
1.35	many*
1.36	230*
1.4	47*
1.5	15*

(b) AG iterations: $\mu_{\text{fac}} = 1$. $*$ =diverged

$L_{\text{fac}} = \mu_{\text{fac}}$	iters
1.1	203
1 2	214
1.3	219
1.35	817
1.36	491*
1.4	52*

(c) AG iterations: $L_{\text{fac}} = \mu_{\text{fac}}$. $*$ =diverged

It's encouraging that really bad settings "fail quickly."

Choosing Parameters Adaptively — or Avoiding Them!

Numerous approaches proposed recently for estimating μ, or designing versions of accelerated gradient that achieve the a convergence rate like the strongly convex case while not requiring μ at all.

- [Nesterov, 2013] detects when μ is too small (by examining the shrinkage of a gradient map), then doubles it and restarts.
- [O'Donoghue and Candès, 2015] consider a different Nesterov accelerated scheme (unified for weakly / strongly convex).
  - They note "cycles" in the behavior and derive an effective restarting heuristic based on this behavior.
  - (Period of the cycles reveals μ/L, but not used explicitly.)
- [Lin and Xiao, 2015] describe an adaptive scheme for estimating μ (based on [Nesterov, 2013]), with restarting, at the cost of a $\log(L/\mu)$ factor in the complexity.
- [Fercoq and Qu, 2016] describe a restarted acceleration scheme that uses estimates of μ but does not rely on them being valid.

Stochastic Gradient: Estimating μ

The estimate of μ is also critical in a strongly convex variant of stochastic gradient [Nemirovski et al., 2009].

For $f(x) := \mathbb{E}_\xi g(x;\xi)$, define

$$x^{k+1} = x^k - \alpha_k g(x^k, \xi_k), \quad \alpha_k = 1/(\mu k).$$

Convergence is

$$\mathbb{E}[\|x^k - x^*\|^2] = O(1/k).$$

But if μ is overestimated (so that steps are too short), convergence is dramatically slower. [Nemirovski et al., 2009] example:

$$f(x) = x^2/10, \quad \text{so that } \mu = .2,$$

but take steps $\alpha_k = (1/k)$ rather than $5/k$, with steps along $-\nabla f(x^k)$ (no noise). Have $x^k > .8k^{-0.2}$.

Line-Search Methods: Wolfe Conditions

Unconstrained minimization of smooth nonconvex function: $\min_x f(x)$.

Line-search methods: At each iterate x, choose search direction p that gives *descent*: $p^T \nabla f(x) < 0$.

Set $x \leftarrow x + \alpha p$, where steplength $\alpha > 0$ satisfies Wolfe conditions:

$$
\begin{aligned}
f(x + \alpha p) &\le f(x) + c_1 \alpha \nabla f(x)^T p, && \text{(not too long)} && (1)\\
|\nabla f(x + \alpha p)^T p| &\le -c_2 \nabla f(x)^T p, && \text{(not too short)} && (2)
\end{aligned}
$$

where $0 < c_1 < c_2 < 1$.

Use specialized one-dimensional searches (based on bracketing and interpolation) to find α satisfying these conditions. Typically require just 2-4 function/gradient evaluations per search.

Folklore choices are $c_1 = 10^{-3}$, $c_2 = .7$. Are these really the best?

Line-Search Methods: Backtracking

Backtracking: Decrease α repeatedly by a constant factor until a sufficient decrease condition is satisfied.

Given $c_1 \in (0,1)$, $\alpha_{\max} > 0$, and $\rho \in (0,1)$, choose α to be the first value in $\bar{\alpha}, \rho\bar{\alpha}, \rho^2\bar{\alpha}, \dots$, that satisfies

$$f(x + \alpha p) \le f(x) + c_1 \alpha \nabla f(x)^T p,$$

where $\bar{\alpha}$ is a first guess. (Set $\bar{\alpha} = \alpha_- / \rho$, where α_- is the successful step from the previous iteration.)

Folklore values: $\rho = .5$, $c_1 = 10^{-3}$.

Performance Profiles

Moré (2007) studied choices of (c_1, c_2) using performance profiles. New experiments (due to Clément Royer)

- Search directions p chosen by steepest descent and nonlinear conjugate gradient.
- Battery of about 204 CUTEst test problems, dimensions $n = 2$-1000.
- Searches over combinations of (c_1, c_2) for line-search, and (c_1, ρ) for backtracking:

$$c_1 = 10^{-10}, \dots, 10^{-1}; \quad c_2 = .5, .75, .8, .9, .95; \quad \rho = .3, .5, .7.$$

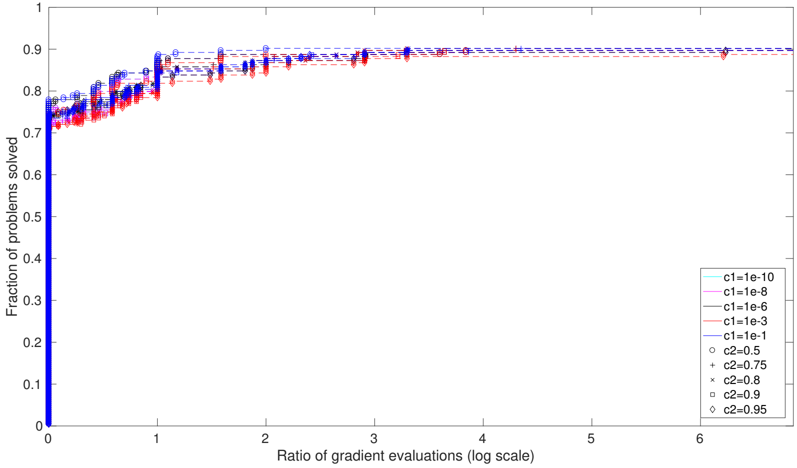
- Keep track of CPU time, function evaluations, gradient evaluations.

For most of the 204 problems, the # gradient evaluations and CPU time are insensitive to the parameters, for the bracketing/zoom approaches.

Most plots *don't include problems in the plots for which all solvers give similar results!*

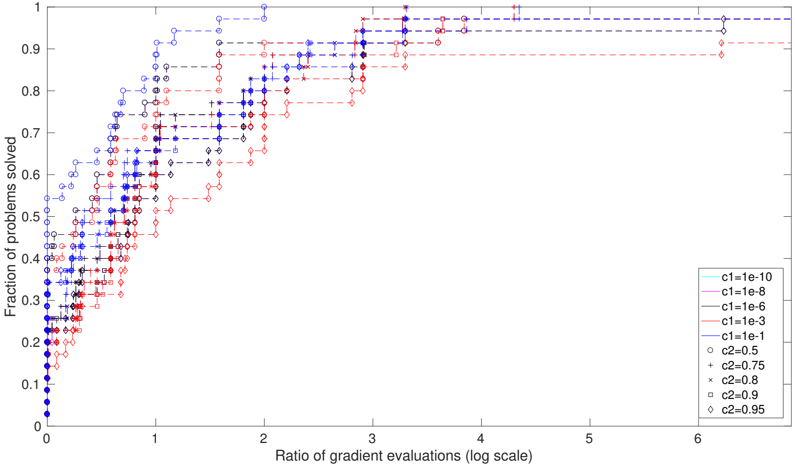
We show mostly plots for # gradient evaluations.

Steepest Descent: Bracketing/Zoom: All Problems



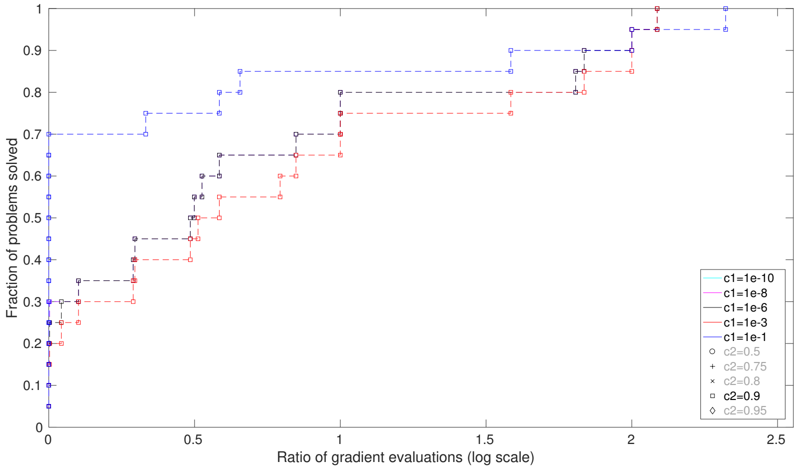
All 204 problems: Most have similar performance! Each code fails to solve about 10% of the problems.

Steepest Descent: Bracketing/Zoom: Distinctive Cases



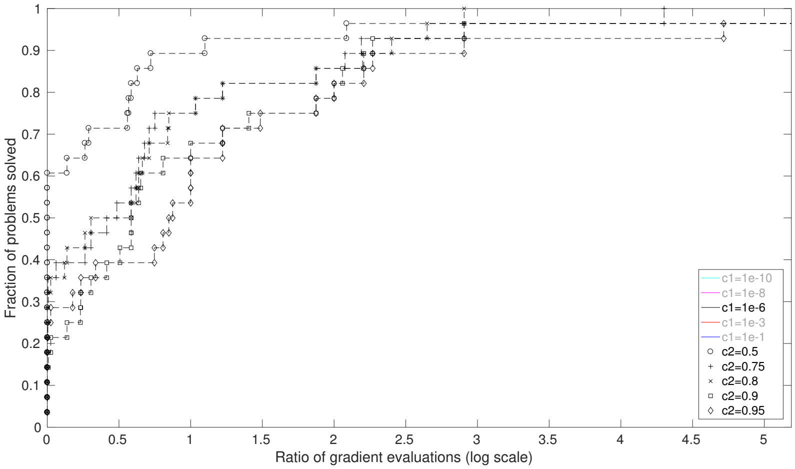
35/204 distinctive cases: Stricter Wolfe values preferred (smaller c_2, larger c_1).

Steepest Descent: Bracketing/Zoom: $c_2 = 0.9$



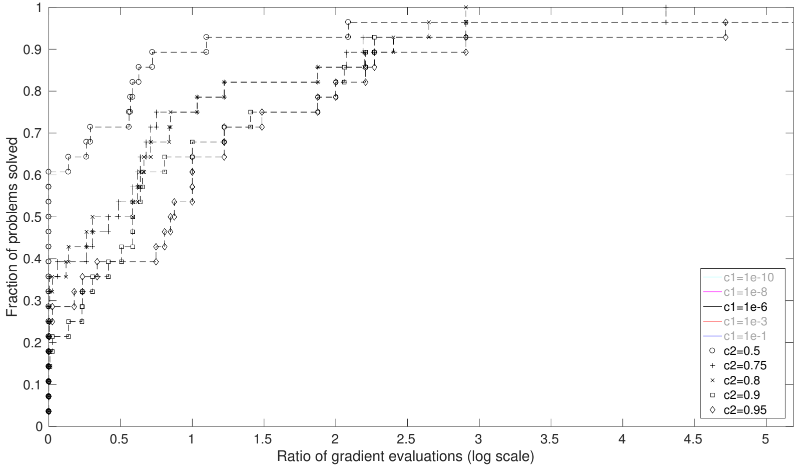
20/204 distinctive cases: Stricter (larger) c_1 values preferred.

Steepest Descent: Bracketing/Zoom: $c_1 = 10^{-6}$



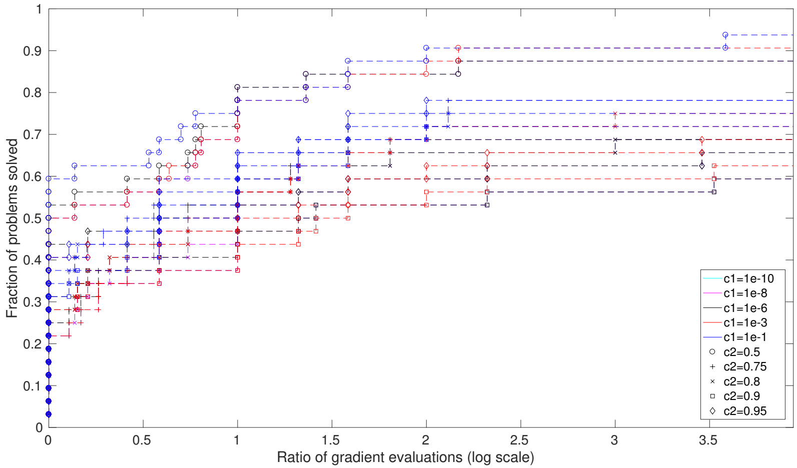
28/204 distinctive cases: Stricter (smaller) c_2 values preferred.

Nonlinear CG: Bracketing/Zoom: All Distinctive Cases



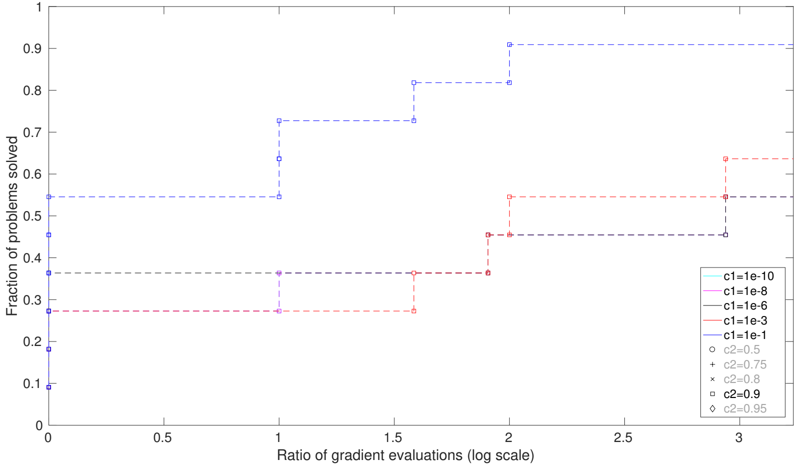
32/204 distinctive cases: Stricter c_1 and c_2 slightly preferred. See divergence in reliability between different param settings.

Nonlinear CG: Bracketing/Zoom: All Distinctive Cases



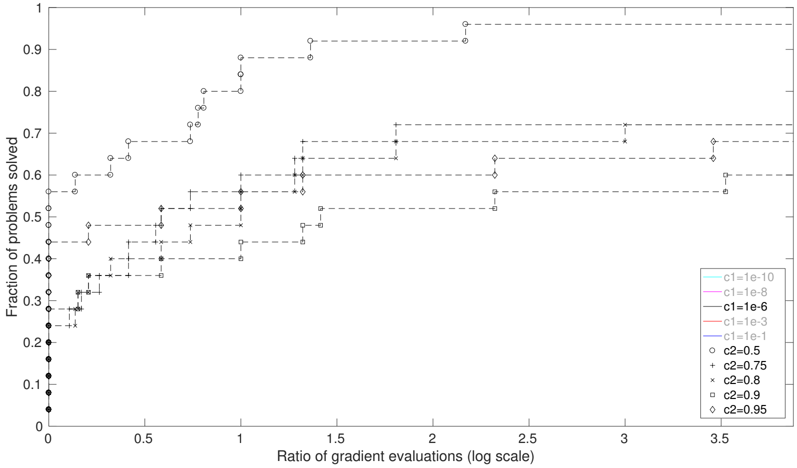
32/204 distinctive cases: Stricter c_1 and c_2 slightly preferred. Divergence in reliability between different parameter settings.

Nonlinear CG: Bracketing/Zoom: $c_2 = .9$



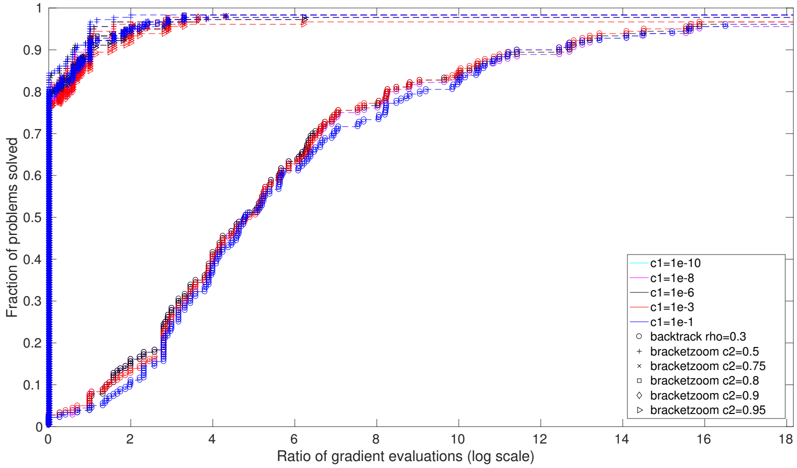
Only 15/204 distinctive cases: Stricter c_1 better on these few cases.

Nonlinear CG: Bracketing/Zoom: $c_1 = 10^{-6}$



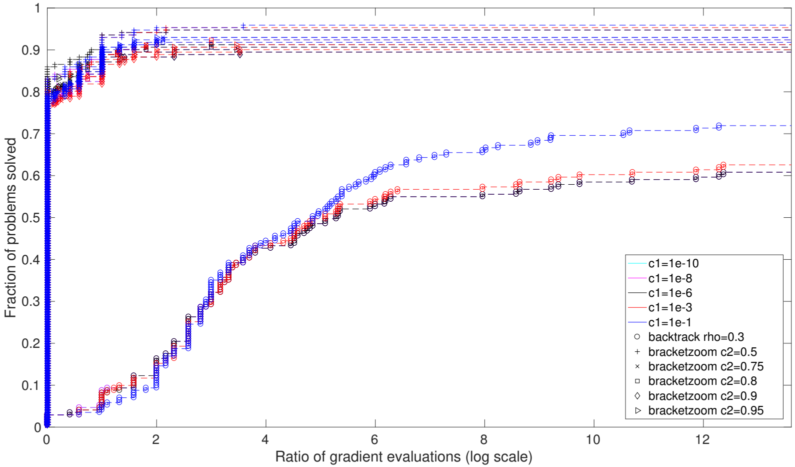
Only 15/204 distinctive cases: Stricter c_2 is better.

Steepest Descent: Bracketing/Zoom vs Backtracking



180/204 distinct problems. Used $\rho = 0.3$ for backtracking — best but clearly inferior. Note insensitivity of B/Z performance to parameters.

Nonlinear CG: Bracketing/Zoom vs Backtracking



171/204 distinct problems. Backtracking still inferior. Note again that reliability of B/Z is somewhat sensitive to parameters.

Line-Search Methods: Notes

- Bracketing/Zoom is better than Backtracking (at least this implementation).
- Not sensitive to parameters on 80% of problems.
- Best parameter settings are similar, for steepest descent and nonlinear conjugate gradient.
- Reliability (eventual termination) is more sensitive to parameters in CG than steepest descent.
- On the "distinctive" problems, preferred Wolfe parameters were slightly more strict than the conventional wisdom allowed.

Continuation in Regularized Optimization

Example: ℓ_1 regularization (LASSO, compressed sensing):

$$x(\lambda) := \arg\min_x \frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1. \tag{3}$$

Many first-order methods proposed (SpaRSA, FISTA, etc) with long antecedents.

- The value of λ impacts the practical difficulty of solving (3).
- Often there is statistical guidance for choosing λ (e.g. based on distribution of errors in y). But usually interested in a range of λ values.
- Solution for large λ is trivial: $\lambda \ge \|A^T y\|_\infty \;\Rightarrow\; x(\lambda) = 0$.

Suggests a continuation heuristic (e.g. [Wright et al., 2009]): Given a target value $\bar{\lambda}$:

- Start with large value $\lambda = \lambda^0$ and solve for $x(\lambda)$;
- Decrease λ by some heuristic; re-solve for $x(\lambda)$, using the previous solution as a starting point.
- Repeat until $\lambda = \bar{\lambda}$.

Continuation for ℓ_2-ℓ_1

Various heuristics tried:

- Reduce λ by a constant factor;
- Adapt the factor according to the number of iterations required to solve for $x(\lambda)$ at the previous λ. Even consider backtracking — increasing λ again.

Continuation improves this class of methods greatly, and performance can be very sensitive to the choice of continuation strategy / parameter. Poor choices can lead to gross inefficiency, or failure.

AFAIK, no rigorous study performed.

Some complexity analysis of continuation strategies done e.g. by [Xiao and Zhang, 2012].

Conclusions

- Good parameter choices are crucial in many optimization contexts.
- In some cases there are > 10 important parameters.
- Parameter selection in optimization is often guided by folklore / accumulated wisdom.
- Systematic procedures for selecting parameters and heuristics still not used much — perhaps should be.
- Choice of test problem batteries is crucial — danger of overfitting.

Thanks to Jorge Moré, Clément Royer.

FIN

References I

Czyzyk, J., Mehrotra, S., Wagner, M., and Wright, S. J. (1997).
PCx User Guide (Version 1.1).
Technical Report OTC 96/01, Optimization Technology Center, Argonne National Laboratory and Northwestern University.
Revised November 1997.

Dolan, E. D. and Moré, J. J. (2002).
Benchmarking optimization software with performance profiles.
Mathematical Programming, Series A, 91(2):201–213.

Fercoq, O. and Qu, Z. (2016).
Restarting accelerated gradient methods with a rough strong convexity estimate.
Technical Report arXiv:1609.07358, LTCI, CNRS, Telecom ParisTech.

Gondzio, J. (1996).
Multiple centrality corrections in a primal-dual method for linear programming.
Computational Optimization and Applications, 6:137–156.

Lin, Q. and Xiao, L. (2015).
An adaptive accelerated proximate gradient method and its homotopy continuation for sparse optimization.
Computational Optimization and Applications, 60:633–674.

References II

Lyness, J. N. and Kaganove, J. J. (1977).
A technique for comparing automatic quadrature routines.
The Computer Journal, 20(2):170–177.

Mehrotra, S. (1992).
On the implementation of a primal-dual interior point method.
SIAM Journal on Optimization, 2:575–601.

Moré, J. J. (2007).
Stalking performance in optimization algorithms.
Talk at "Conference on Combinatorics and Optimization," University of Waterloo.

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.

Nesterov, Y. (2013).
Gradient methods for minimizing composite functions.
Mathematical Programming, Series B, 140:125–161.

O'Donoghue, B. and Candès, E. (2015).
Adaptive restart for accelerated gradient schemes.
Foundations of Computational Mathematics, 15:715–732.

References III

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

Wright, S. J. (1999).
Modified Cholesky factorizations in interior-point algorithms for linear programming.
SIAM Journal on Optimization, 9:1159–1191.

Wright, S. J., Nowak, R. D., and Figueiredo, M. A. T. (2009).
Sparse reconstruction by separable approximation.
IEEE Transactions on Signal Processing, 57:2479–2493.

Xiao, L. and Zhang, T. (2012).
A proximal-gradient homotopy method for the sparse least-squares problem.
Technical Report arXiv:1203:3002v1, Microsoft Research.