Convex Optimization

Convex Optimization

Stephen Boyd

Department of Electrical Engineering
Stanford University

Lieven Vandenberghe

Electrical Engineering Department
University of California, Los Angeles

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Published in the United States of America by Cambridge University Press, New York

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521833783

First published 2004
15th printing 2014

Printed in the United Kingdom by T.J. International Ltd, Padstow, Cornwall

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing-in-Publication data

Boyd, Stephen P.
Convex Optimization / Stephen Boyd & Lieven Vandenberghe
p. cm.
Includes bibliographical references and index.
ISBN 0 521 83378 7
1. Mathematical optimization. 2. Convex functions. I. Vandenberghe, Lieven. II. Title.

QA402.5.B69 2004
519.6–dc22 2003063284

ISBN 978-0-521-83378-3 Hardback

Contents

Preface

This book is about *convex optimization*, a special class of mathematical optimization problems, which includes least-squares and linear programming problems. It is well known that least-squares and linear programming problems have a fairly complete theory, arise in a variety of applications, and can be solved numerically very efficiently. The basic point of this book is that the same can be said for the larger class of convex optimization problems.

While the mathematics of convex optimization has been studied for about a century, several related recent developments have stimulated new interest in the topic. The first is the recognition that interior-point methods, developed in the 1980s to solve linear programming problems, can be used to solve convex optimization problems as well. These new methods allow us to solve certain new classes of convex optimization problems, such as semidefinite programs and second-order cone programs, almost as easily as linear programs.

The second development is the discovery that convex optimization problems (beyond least-squares and linear programs) are more prevalent in practice than was previously thought. Since 1990 many applications have been discovered in areas such as automatic control systems, estimation and signal processing, communications and networks, electronic circuit design, data analysis and modeling, statistics, and finance. Convex optimization has also found wide application in combinatorial optimization and global optimization, where it is used to find bounds on the optimal value, as well as approximate solutions. We believe that many other applications of convex optimization are still waiting to be discovered.

There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that the problem can then be solved, very reliably and efficiently, using interior-point methods or other special methods for convex optimization. These solution methods are reliable enough to be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system. There are also theoretical or conceptual advantages of formulating a problem as a convex optimization problem. The associated dual problem, for example, often has an interesting interpretation in terms of the original problem, and sometimes leads to an efficient or distributed method for solving it.

We think that convex optimization is an important enough topic that everyone who uses computational mathematics should know at least a little bit about it. In our opinion, convex optimization is a natural next topic after advanced linear algebra (topics like least-squares, singular values), and linear programming.

Goal of this book

For many general purpose optimization methods, the typical approach is to just try out the method on the problem to be solved. The full benefits of convex optimization, in contrast, only come when the problem is known ahead of time to be convex. Of course, many optimization problems are not convex, and it can be difficult to recognize the ones that are, or to reformulate a problem so that it is convex.

> *Our main goal is to help the reader develop a working knowledge of convex optimization, i.e., to develop the skills and background needed to recognize, formulate, and solve convex optimization problems.*

Developing a working knowledge of convex optimization can be mathematically demanding, especially for the reader interested primarily in applications. In our experience (mostly with graduate students in electrical engineering and computer science), the investment often pays off well, and sometimes very well.

There are several books on linear programming, and general nonlinear programming, that focus on problem formulation, modeling, and applications. Several other books cover the theory of convex optimization, or interior-point methods and their complexity analysis. This book is meant to be something in between, a book on general convex optimization that focuses on problem formulation and modeling.

We should also mention what this book is *not*. It is not a text primarily about convex analysis, or the mathematics of convex optimization; several existing texts cover these topics well. Nor is the book a survey of algorithms for convex optimization. Instead we have chosen just a few good algorithms, and describe only simple, stylized versions of them (which, however, do work well in practice). We make no attempt to cover the most recent state of the art in interior-point (or other) methods for solving convex problems. Our coverage of numerical implementation issues is also highly simplified, but we feel that it is adequate for the potential user to develop working implementations, and we do cover, in some detail, techniques for exploiting structure to improve the efficiency of the methods. We also do not cover, in more than a simplified way, the complexity theory of the algorithms we describe. We do, however, give an introduction to the important ideas of self-concordance and complexity analysis for interior-point methods.

Audience

This book is meant for the researcher, scientist, or engineer who uses mathematical optimization, or more generally, computational mathematics. This includes, naturally, those working directly in optimization and operations research, and also many others who use optimization, in fields like computer science, economics, finance, statistics, data mining, and many fields of science and engineering. Our primary focus is on the latter group, the potential *users* of convex optimization, and not the (less numerous) experts in the field of convex optimization.

The only background required of the reader is a good knowledge of advanced calculus and linear algebra. If the reader has seen basic mathematical analysis (*e.g.*, norms, convergence, elementary topology), and basic probability theory, he or she should be able to follow every argument and discussion in the book. We hope that readers who have not seen analysis and probability, however, can still get all of the essential ideas and important points. Prior exposure to numerical computing or optimization is not needed, since we develop all of the needed material from these areas in the text or appendices.

Using this book in courses

We hope that this book will be useful as the primary or alternate textbook for several types of courses. Since 1995 we have been using drafts of this book for graduate courses on linear, nonlinear, and convex optimization (with engineering applications) at Stanford and UCLA. We are able to cover most of the material, though not in detail, in a one quarter graduate course. A one semester course allows for a more leisurely pace, more applications, more detailed treatment of theory, and perhaps a short student project. A two quarter sequence allows an expanded treatment of the more basic topics such as linear and quadratic programming (which are very useful for the applications oriented student), or a more substantial student project.

This book can also be used as a reference or alternate text for a more traditional course on linear and nonlinear optimization, or a course on control systems (or other applications area), that includes some coverage of convex optimization. As the secondary text in a more theoretically oriented course on convex optimization, it can be used as a source of simple practical examples.

Acknowledgments

We have been developing the material for this book for almost a decade. Over the years we have benefited from feedback and suggestions from many people, including our own graduate students, students in our courses, and our colleagues at Stanford, UCLA, and elsewhere. Unfortunately, space limitations and shoddy record keeping do not allow us to name everyone who has contributed. However, we wish to particularly thank A. Aggarwal, V. Balakrishnan, A. Bernard, B. Bray, R. Cottle, A. d'Aspremont, J. Dahl, J. Dattorro, D. Donoho, J. Doyle, L. El Ghaoui, P. Glynn, M. Grant, A. Hansson, T. Hastie, A. Lewis, M. Lobo, Z.-Q. Luo, M. Mesbahi, W. Naylor, P. Parrilo, I. Pressman, R. Tibshirani, B. Van Roy, L. Xiao, and Y. Ye. J. Jalden and A. d'Aspremont contributed the time-frequency analysis example in §6.5.4, and the consumer preference bounding example in §6.5.5, respectively. P. Parrilo suggested exercises 4.4 and 4.56. Newer printings benefited greatly from Igal Sason's meticulous reading of the book.

We want to single out two others for special acknowledgment. Arkadi Nemirovski incited our original interest in convex optimization, and encouraged us to write this book. We also want to thank Kishan Baheti for playing a critical role in the development of this book. In 1994 he encouraged us to apply for a National Science Foundation combined research and curriculum development grant, on convex optimization with engineering applications, and this book is a direct (if delayed) consequence.

Stephen Boyd — *Stanford, California*
Lieven Vandenberghe — *Los Angeles, California*

Chapter 1

Introduction

In this introduction we give an overview of mathematical optimization, focusing on the special role of convex optimization. The concepts introduced informally here will be covered in later chapters, with more care and technical detail.

1.1 Mathematical optimization

A *mathematical optimization problem*, or just *optimization problem*, has the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m. \end{array} \tag{1.1}$$

Here the vector $x = (x_1, \ldots, x_n)$ is the *optimization variable* of the problem, the function $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the *objective* function, the functions $f_i : \mathbf{R}^n \to \mathbf{R}$, $i = 1, \ldots, m$, are the (inequality) *constraint functions*, and the constants $b_1, \ldots, b_m$ are the limits, or bounds, for the constraints. A vector $x^\star$ is called *optimal*, or a *solution* of the problem (1.1), if it has the smallest objective value among all vectors that satisfy the constraints: for any z with $f_1(z) \leq b_1, \ldots, f_m(z) \leq b_m$, we have $f_0(z) \geq f_0(x^\star)$.

We generally consider families or classes of optimization problems, characterized by particular forms of the objective and constraint functions. As an important example, the optimization problem (1.1) is called a *linear program* if the objective and constraint functions $f_0, \ldots, f_m$ are linear, *i.e.*, satisfy

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y) \tag{1.2}$$

for all x, $y \in \mathbf{R}^n$ and all α, $\beta \in \mathbf{R}$. If the optimization problem is not linear, it is called a *nonlinear program*.

This book is about a class of optimization problems called *convex optimization problems*. A convex optimization problem is one in which the objective and constraint functions are convex, which means they satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{1.3}$$

for all $x, y \in \mathbf{R}^n$ and all $\alpha, \beta \in \mathbf{R}$ with $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. Comparing (1.3) and (1.2), we see that convexity is more general than linearity: inequality replaces the more restrictive equality, and the inequality must hold only for certain values of α and β. Since any linear program is therefore a convex optimization problem, we can consider convex optimization to be a generalization of linear programming.

1.1.1 Applications

The optimization problem (1.1) is an abstraction of the problem of making the best possible choice of a vector in $\mathbf{R}^n$ from a set of candidate choices. The variable x represents the choice made; the constraints $f_i(x) \leq b_i$ represent firm requirements or specifications that limit the possible choices, and the objective value $f_0(x)$ represents the cost of choosing x. (We can also think of $-f_0(x)$ as representing the value, or utility, of choosing x.) A solution of the optimization problem (1.1) corresponds to a choice that has minimum cost (or maximum utility), among all choices that meet the firm requirements.

In *portfolio optimization*, for example, we seek the best way to invest some capital in a set of n assets. The variable x_i represents the investment in the ith asset, so the vector $x \in \mathbf{R}^n$ describes the overall portfolio allocation across the set of assets. The constraints might represent a limit on the budget (*i.e.*, a limit on the total amount to be invested), the requirement that investments are nonnegative (assuming short positions are not allowed), and a minimum acceptable value of expected return for the whole portfolio. The objective or cost function might be a measure of the overall risk or variance of the portfolio return. In this case, the optimization problem (1.1) corresponds to choosing a portfolio allocation that minimizes risk, among all possible allocations that meet the firm requirements.

Another example is *device sizing* in electronic design, which is the task of choosing the width and length of each device in an electronic circuit. Here the variables represent the widths and lengths of the devices. The constraints represent a variety of engineering requirements, such as limits on the device sizes imposed by the manufacturing process, timing requirements that ensure that the circuit can operate reliably at a specified speed, and a limit on the total area of the circuit. A common objective in a device sizing problem is the total power consumed by the circuit. The optimization problem (1.1) is to find the device sizes that satisfy the design requirements (on manufacturability, timing, and area) and are most power efficient.

In *data fitting*, the task is to find a model, from a family of potential models, that best fits some observed data and prior information. Here the variables are the parameters in the model, and the constraints can represent prior information or required limits on the parameters (such as nonnegativity). The objective function might be a measure of misfit or prediction error between the observed data and the values predicted by the model, or a statistical measure of the unlikeliness or implausibility of the parameter values. The optimization problem (1.1) is to find the model parameter values that are consistent with the prior information, and give the smallest misfit or prediction error with the observed data (or, in a statistical

x
its
ep-
g the
corre-
choices

est some
in the ith
ss the set of
limit on the
e nonnegative
ptable value of
nction might be
rn. In this case,
olio allocation that
rm requirements.
ch is the task of choos-
cuit. Here the variables
nstraints represent a va-
e device sizes imposed by
ensure that the circuit can
ie total area of the circuit. A
total power consumed by the
the device sizes that satisfy the
g, and area) and are most power

, from a family of potential models,
nformation. Here the variables are the
ints can represent prior information or
as nonnegativity). The objective function
ction error between the observed data and
r a statistical measure of the unlikeliness or
ues. The optimization problem (1.1) is to find
re consistent with the prior information, and give
error with the observed data (or, in a statistical

framework, are most likely).

An amazing variety of practical problems involving decision making (or system design, analysis, and operation) can be cast in the form of a mathematical optimization problem, or some variation such as a multicriterion optimization problem. Indeed, mathematical optimization has become an important tool in many areas. It is widely used in engineering, in electronic design automation, automatic control systems, and optimal design problems arising in civil, chemical, mechanical, and aerospace engineering. Optimization is used for problems arising in network design and operation, finance, supply chain management, scheduling, and many other areas. The list of applications is still steadily expanding.

For most of these applications, mathematical optimization is used as an aid to a human decision maker, system designer, or system operator, who supervises the process, checks the results, and modifies the problem (or the solution approach) when necessary. This human decision maker also carries out any actions suggested by the optimization problem, e.g., buying or selling assets to achieve the optima portfolio.

A relatively recent phenomenon opens the possibility of many other applicati for mathematical optimization. With the proliferation of computers embedd products, we have seen a rapid growth in *embedded optimization*. In thes bedded applications, optimization is used to automatically make real-time and even carry out the associated actions, with no (or little) human interv oversight. In some application areas, this blending of traditional automa systems and embedded optimization is well under way; in others, it is ing. Embedded real-time optimization raises some new challenges: i it requires solution methods that are extremely reliable, and solve predictable amount of time (and memory).

1.1.2 Solving optimization problems

A *solution method* for a class of optimization problems is ar putes a solution of the problem (to some given accuracy), giv from the class, i.e., an *instance* of the problem. Since the has gone into developing algorithms for solving various cla lems, analyzing their properties, and developing good The effectiveness of these algorithms, i.e., our ability to lem (1.1), varies considerably, and depends on factor of the objective and constraint functions, how many are, and special structure, such as *sparsity*. (A pro function depends on only a small number of the

Even when the objective and constraint fu polynomials) the general optimization problem Approaches to the general problem therefore i as very long computation time, or the possi of these methods are discussed in §1.4.

There are, however, some important optimization problems are difficult to effective algorithms that can reliably solve even large problems, with hundreds or thousands of variables and constraints. Two important and well known examples, described in §1.2 below (and in detail in chapter 4), are least-squares problems and linear programs. It is less well known that convex optimization is another exception to the rule: Like least-squares or linear programming, there are very effective algorithms that can reliably and efficiently solve even large convex problems.

1.2 Least-squares and linear programming

In this section we describe two very widely known and used special subclasses of convex optimization: least-squares and linear programming. (A complete technical treatment of these problems will be given in chapter 4.)

1.2.1 Least-squares problems

A *least-squares* problem is an optimization problem with no constraints (*i.e.*, $m = 0$) and an objective which is a sum of squares of terms of the form $a_i^T x - b_i$:

$$\text{minimize} \quad f_0(x) = \|Ax - b\|_2^2 = \textstyle\sum_{i=1}^k (a_i^T x - b_i)^2. \tag{1.4}$$

Here $A \in \mathbf{R}^{k \times n}$ (with $k \geq n$), a_i^T are the rows of A, and the vector $x \in \mathbf{R}^n$ is the optimization variable.

Solving least-squares problems

The solution of a least-squares problem (1.4) can be reduced to solving a set of linear equations,

$$(A^T A)x = A^T b,$$

so we have the analytical solution $x = (A^T A)^{-1} A^T b$. For least-squares problems we have good algorithms (and software implementations) for solving the problem to high accuracy, with very high reliability. The least-squares problem can be solved in a time approximately proportional to $n^2 k$, with a known constant. A current desktop computer can solve a least-squares problem with hundreds of variables, and thousands of terms, in a few seconds; more powerful computers, of course, can solve larger problems, or the same size problems, faster. (Moreover, these solution times will decrease exponentially in the future, according to Moore's law.) Algorithms and software for solving least-squares problems are reliable enough for embedded optimization.

In many cases we can solve even larger least-squares problems, by exploiting some special structure in the coefficient matrix A. Suppose, for example, that the matrix A is *sparse*, which means that it has far fewer than kn nonzero entries. By exploiting sparsity, we can usually solve the least-squares problem much faster than order $n^2 k$. A current desktop computer can solve a sparse least-squares problem with tens of thousands of variables, and hundreds of thousands of terms, in around a minute (although this depends on the particular sparsity pattern).

For extremely large problems (say, with millions of variables), or for problems with exacting real-time computing requirements, solving a least-squares problem can be a challenge. But in the vast majority of cases, we can say that existing methods are very effective, and extremely reliable. Indeed, we can say that solving least-squares problems (that are not on the boundary of what is currently achievable) is a (mature) *technology*, that can be reliably used by many people who do not know, and do not need to know, the details.

Using least-squares

The least-squares problem is the basis for regression analysis, optimal control, and many parameter estimation and data fitting methods. It has a number of statistical interpretations, *e.g.*, as maximum likelihood estimation of a vector x, given linear measurements corrupted by Gaussian measurement errors.

Recognizing an optimization problem as a least-squares problem is straightforward; we only need to verify that the objective is a quadratic function (and then test whether the associated quadratic form is positive semidefinite). While the basic least-squares problem has a simple fixed form, several standard techniques are used to increase its flexibility in applications.

In *weighted least-squares*, the weighted least-squares cost

$$\sum_{i=1}^{k} w_i(a_i^T x - b_i)^2,$$

where $w_1, \ldots, w_k$ are positive, is minimized. (This problem is readily cast and solved as a standard least-squares problem.) Here the weights w_i are chosen to reflect differing levels of concern about the sizes of the terms $a_i^T x - b_i$, or simply to influence the solution. In a statistical setting, weighted least-squares arises in estimation of a vector x, given linear measurements corrupted by errors with unequal variances.

Another technique in least-squares is *regularization*, in which extra terms are added to the cost function. In the simplest case, a positive multiple of the sum of squares of the variables is added to the cost function:

$$\sum_{i=1}^{k}(a_i^T x - b_i)^2 + \rho \sum_{i=1}^{n} x_i^2,$$

where $\rho > 0$. (This problem too can be formulated as a standard least-squares problem.) The extra terms penalize large values of x, and result in a sensible solution in cases when minimizing the first sum only does not. The parameter ρ is chosen by the user to give the right trade-off between making the original objective function $\sum_{i=1}^{k}(a_i^T x - b_i)^2$ small, while keeping $\sum_{i=1}^{n} x_i^2$ not too big. Regularization comes up in statistical estimation when the vector x to be estimated is given a prior distribution.

Weighted least-squares and regularization are covered in chapter 6; their statistical interpretations are given in chapter 7.

parameters that specify the objective and constraint fu

Solving linear programs

There is no simple analytical formula for the solution
is for a least-squares problem), but there are a variety
solving them, including Dantzig's simplex method, an
point methods described later in this book. While we ca
of arithmetic operations required to solve a linear pro
squares), we can establish rigorous bounds on the num
to solve a linear program, to a given accuracy, using an i
complexity in practice is order n^2m (assuming $m \geq n$) b
less well characterized than for least-squares. These algo
although perhaps not quite as reliable as methods for leas
solve problems with hundreds of variables and thousands
desktop computer, in a matter of seconds. If the problem
other exploitable structure, we can often solve problems w
thousands of variables and constraints.

As with least-squares problems, it is still a challenge t
linear programs, or to solve linear programs with exacting r
quirements. But, like least-squares, we can say that solving
is a mature technology. Linear programming solvers can be
many tools and applications.

Using linear programming

Some applications lead directly to linear programs in the f
several other standard forms. In many other cases the origina
lem does not have a standard linear program form, but can b
equivalent linear program (and then, of course, solved) using t
detail in chapter 4.

One other important distinction is that the objective function in the Chebyshev approximation problem (1.6) is not differentiable; the objective in the least-squares problem (1.4) is quadratic, and therefore differentiable.

The Chebyshev approximation problem (1.6) can be solved by solving the linear program

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & a_i^T x - t \leq b_i, \quad i = 1, \ldots, k \\ & -a_i^T x - t \leq -b_i, \quad i = 1, \ldots, k, \end{array} \tag{1.7}$$

with variables $x \in \mathbf{R}^n$ and $t \in \mathbf{R}$. (The details will be given in chapter 6.) Since linear programs are readily solved, the Chebyshev approximation problem is therefore readily solved.

Anyone with a working knowledge of linear programming would recognize the Chebyshev approximation problem (1.6) as one that can be reduced to a linear program. For those without this background, though, it might not be obvious that the Chebyshev approximation problem (1.6), with its nondifferentiable objective, can be formulated and solved as a linear program.

While recognizing problems that can be reduced to linear programs is more involved than recognizing a least-squares problem, it is a skill that is readily acquired, since only a few standard tricks are used. The task can even be partially automated; some software systems for specifying and solving optimization problems can automatically recognize (some) problems that can be reformulated as linear programs.

1.3 Convex optimization

A convex optimization problem is one of the form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{array} \tag{1.8}$$

where the functions $f_0, \ldots, f_m : \mathbf{R}^n \to \mathbf{R}$ are convex, *i.e.*, satisfy

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

for all x, $y \in \mathbf{R}^n$ and all α, $\beta \in \mathbf{R}$ with $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. The least-squares problem (1.4) and linear programming problem (1.5) are both special cases of the general convex optimization problem (1.8).

1.3.1 Solving convex optimization problems

There is in general no analytical formula for the solution of convex optimization problems, but (as with linear programming problems) there are very effective methods for solving them. Interior-point methods work very well in practice, and in some cases can be proved to solve the problem to a specified accuracy with a number of operations that does not exceed a polynomial of the problem dimensions. (This is covered in chapter 11.)

We will see that interior-point methods can solve the problem (1.8) in a number of steps or iterations that is almost always in the range between 10 and 100. Ignoring any structure in the problem (such as sparsity), each step requires on the order of

$$\max\{n^3, n^2m, F\}$$

operations, where F is the cost of evaluating the first and second derivatives of the objective and constraint functions $f_0, \dots, f_m$.

Like methods for solving linear programs, these interior-point methods are quite reliable. We can easily solve problems with hundreds of variables and thousands of constraints on a current desktop computer, in at most a few tens of seconds. By exploiting problem structure (such as sparsity), we can solve far larger problems, with many thousands of variables and constraints.

We cannot yet claim that solving general convex optimization problems is a mature technology, like solving least-squares or linear programming problems. Research on interior-point methods for general nonlinear convex optimization is still a very active research area, and no consensus has emerged yet as to what the best method or methods are. But it is reasonable to expect that solving general convex optimization problems will become a technology within a few years. And for some subclasses of convex optimization problems, for example second-order cone programming or geometric programming (studied in detail in chapter 4), it is fair to say that interior-point methods are approaching a technology.

1.3.2 Using convex optimization

Using convex optimization is, at least conceptually, very much like using least-squares or linear programming. If we can formulate a problem as a convex optimization problem, then we can solve it efficiently, just as we can solve a least-squares problem efficiently. With only a bit of exaggeration, we can say that, if you formulate a practical problem as a convex optimization problem, then you have solved the original problem.

There are also some important differences. Recognizing a least-squares problem is straightforward, but recognizing a convex function can be difficult. In addition, there are many more tricks for transforming convex problems than for transforming linear programs. Recognizing convex optimization problems, or those that can be transformed to convex optimization problems, can therefore be challenging. The main goal of this book is to give the reader the background needed to do this. Once the skill of recognizing or formulating convex optimization problems is developed, you will find that surprisingly many problems can be solved via convex optimization.

The challenge, and art, in using convex optimization is in recognizing and formulating the problem. Once this formulation is done, solving the problem is, like least-squares or linear programming, (almost) technology.

Nonlinear optimization (or nonlinear programming) is the term used to describe an optimization problem when the objective or constraint functions are not linear, but not known to be convex. Sadly, there are no effective methods for solving the general nonlinear programming problem (1.1). Even simple looking problems with as few as ten variables can be extremely challenging, while problems with a few hundreds of variables can be intractable. Methods for the general nonlinear programming problem therefore take several different approaches, each of which involves some compromise.

1.4.1 Local optimization

In *local optimization*, the compromise is to give up seeking the optimal x, which minimizes the objective over all feasible points. Instead we seek a point that is only locally optimal, which means that it minimizes the objective function among feasible points that are near it, but is not guaranteed to have a lower objective value than all other feasible points. A large fraction of the research on general nonlinear programming has focused on methods for local optimization, which as a consequence are well developed.

Local optimization methods can be fast, can handle large-scale problems, and are widely applicable, since they only require differentiability of the objective and constraint functions. As a result, local optimization methods are widely used in applications where there is value in finding a good point, if not the very best. In an engineering design application, for example, local optimization can be used to improve the performance of a design originally obtained by manual, or other, design methods.

There are several disadvantages of local optimization methods, beyond (possibly) not finding the true, globally optimal solution. The methods require an initial guess for the optimization variable. This initial guess or starting point is critical, and can greatly affect the objective value of the local solution obtained. Little information is provided about how far from (globally) optimal the local solution is. Local optimization methods are often sensitive to algorithm parameter values, which may need to be adjusted for a particular problem, or family of problems.

Using a local optimization method is trickier than solving a least-squares problem, linear program, or convex optimization problem. It involves experimenting with the choice of algorithm, adjusting algorithm parameters, and finding a good enough initial guess (when one instance is to be solved) or a method for producing a good enough initial guess (when a family of problems is to be solved). Roughly speaking, local optimization methods are more art than technology. Local optimization is a well developed art, and often very effective, but it is nevertheless an art. In contrast, there is little art involved in solving a least-squares problem or a linear program (except, of course, those on the boundary of what is currently possible).

An interesting comparison can be made between local optimization methods for nonlinear programming, and convex optimization. Since differentiability of the objective and constraint functions is the only requirement for most local optimization methods, formulating a practical problem as a nonlinear optimization problem is relatively straightforward. The art in local optimization is in solving the problem (in the weakened sense of finding a locally optimal point), once it is formulated. In convex optimization these are reversed: The art and challenge is in problem formulation; once a problem is formulated as a convex optimization problem, it is relatively straightforward to solve it.

1.4.2 Global optimization

In *global optimization*, the true global solution of the optimization problem (1.1) is found; the compromise is efficiency. The worst-case complexity of global optimization methods grows exponentially with the problem sizes n and m; the hope is that in practice, for the particular problem instances encountered, the method is far faster. While this favorable situation does occur, it is not typical. Even small problems, with a few tens of variables, can take a very long time (*e.g.*, hours or days) to solve.

Global optimization is used for problems with a small number of variables, where computing time is not critical, and the value of finding the true global solution is very high. One example from engineering design is *worst-case analysis* or *verification* of a high value or safety-critical system. Here the variables represent uncertain parameters, that can vary during manufacturing, or with the environment or operating condition. The objective function is a utility function, *i.e.*, one for which smaller values are worse than larger values, and the constraints represent prior knowledge about the possible parameter values. The optimization problem (1.1) is the problem of finding the *worst-case* values of the parameters. If the worst-case value is acceptable, we can certify the system as safe or reliable (with respect to the parameter variations).

A local optimization method can rapidly find a set of parameter values that is bad, but not guaranteed to be the absolute worst possible. If a local optimization method finds parameter values that yield unacceptable performance, it has succeeded in determining that the system is not reliable. But a local optimization method cannot certify the system as reliable; it can only fail to find bad parameter values. A global optimization method, in contrast, will find the absolute worst values of the parameters, and if the associated performance is acceptable, can certify the system as safe. The cost is computation time, which can be very large, even for a relatively small number of parameters. But it may be worth it in cases where the value of certifying the performance is high, or the cost of being wrong about the reliability or safety is high.

1.4.3 Role of convex optimization in nonconvex problems

In this book we focus primarily on convex optimization problems, and applications that can be reduced to convex optimization problems. But convex optimization also plays an important role in problems that are *not* convex.

Initialization for local optimization

One obvious use is to combine convex optimization with a local optimization method. Starting with a nonconvex problem, we first find an approximate, but convex, formulation of the problem. By solving this approximate problem, which can be done easily and without an initial guess, we obtain the exact solution to the approximate convex problem. This point is then used as the starting point for a local optimization method, applied to the original nonconvex problem.

Convex heuristics for nonconvex optimization

Convex optimization is the basis for several heuristics for solving nonconvex problems. One interesting example we will see is the problem of finding a *sparse* vector x (*i.e.*, one with few nonzero entries) that satisfies some constraints. While this is a difficult combinatorial problem, there are some simple heuristics, based on convex optimization, that often find fairly sparse solutions. (These are described in chapter 6.)

Another broad example is given by *randomized algorithms*, in which an approximate solution to a nonconvex problem is found by drawing some number of candidates from a probability distribution, and taking the best one found as the approximate solution. Now suppose the family of distributions from which we will draw the candidates is parametrized, *e.g.*, by its mean and covariance. We can then pose the question, which of these distributions gives us the smallest expected value of the objective? It turns out that this problem is sometimes a convex problem, and therefore efficiently solved. (See, *e.g.*, exercise 11.23.)

Bounds for global optimization

Many methods for global optimization require a cheaply computable lower bound on the optimal value of the nonconvex problem. Two standard methods for doing this are based on convex optimization. In *relaxation*, each nonconvex constraint is replaced with a looser, but convex, constraint. In *Lagrangian relaxation*, the Lagrangian dual problem (described in chapter 5) is solved. This problem is convex, and provides a lower bound on the optimal value of the nonconvex problem.

1.5 Outline

The book is divided into three main parts, titled *Theory*, *Applications*, and *Algorithms*.

1.5.1 Part I: Theory

In part I, *Theory*, we cover basic definitions, concepts, and results from convex analysis and convex optimization. We make no attempt to be encyclopedic, and skew our selection of topics toward those that we think are useful in recognizing