Computational Method of Optimization

Introduction- Lecture I

Introduction

- Topics to be covered

- Mathematical optimization
- Least-squares and linear programming
- Convex optimization
- Example
- Nonlinear optimization
- Brief history of convex optimization

Source: Convex Optimization — Boyd & Vandenberghe

Mathematical optimization

(mathematical) optimization problem
minimize $f_0(x)$
subject to $f_i(x) \leq b_i$, $i = 1,...,m$

- $x = (x_1, \ldots, x_n)$: optimization variables
- $f_0 : R^n \rightarrow R$: objective function
- $f_i : R^n \rightarrow R$, $i = 1,...,m$: constraint functions
- optimal solution $x^\star$ has smallest value of f_0

among all vectors that satisfy the constraints

You have a variable x with n components. These are loosely called the optimization variable. Colloquially, x_1 through x_n are the variables.

When you make a choice x, you'll be scored on an objective. That's f_0 of x. We'll choose to minimize that. In fact, you ask what if you wanted to maximize it? Then you would minimize the negative of that function, for example. It's convenient to have a canonical form where you minimize instead of maximize.

This will be subject to some constraints.

In the simplest case, we'll just take a bunch of inequalities that's $f_i(x)$ is less than b_i. Here, without any lost generality, the b_i s could be zero, because I could subtract b_i from f_i and call that a new f_i . There'd be no harm.

Example 1

Portfolio optimization
Examples
- variables: amounts invested in different assets
- constraints: budget, max./min. investment per asset, minimum return
- objective: overall risk or return variance

In portfolio optimization, decision variables x_1 through x_n – these might represent the amount you invest in N assets. For example, if x_1 is negative, it means you're shorting that asset, in which case you're borrowing money. x_1 through x_n is a portfolio allocation

There will be a budget constraint. That tells you the amount that you have to invest. There might be maximum and minimum investment per asset. For example, the minimum could be zero.

You might have a maximum amount you're allowed to invest in any asset. Your objective might be something like the overall risk or variance in the portfolio. You'd say here's 500 stocks you can invest in. My mean return absolutely must exceed 11 percent. I want the one with the smallest variance in return. That would be the one with the least risk. This is one where when you solved the optimization problem.

Example 2

Device sizing in electronic circuits
- variables: device widths and lengths
- constraints: manufacturing limits, timing requirements, maximum area
- objective: power consumption

Here, you have a circuit. The variables, for example, might be the device widths and lengths. You have constraints. Some of these are manufacturing limits.

For example, depending on what fabrication facility is making your circuit, you have a limit on how small the length of a gate and a transistor can be. It can't be any smaller than 65 nanometers. That's the smallest it can be.

Performance requirements would be things like timing requirements. You could say this circuit has to clock at 400 megahertz, for example. That tells you that the delay in the circuit has to be less than some number because it has to be done doing whatever it's doing before the next clock cycle comes up. That's a timing requirement.

You might have an area requirement. This portion of the circuit can't be anymore than one millimeter squared. The objective in this case might be power consumption. That would be minimized power consumption.

Example 3

Data fitting
- variables: model parameters
- constraints: prior information, parameter limits
- objective: measure of misfit or prediction error

It's a generic estimation model. In these cases, the x's are things you're going to do. In this case, they're portfolios you're going to hold, and in this case, they will translate into polygons that get sent off to the fabrication facility. Here, it's very interesting, because the x_i's are not actions. They're actually parameters that you're estimating.
Here, you have a model. You can take any kind of data or something like that and you'll have parameters in your model. You want to estimate those parameters. These x_i are not things you're going to do. These are numbers you want to estimate. That's what it is.

You have constraints. For example, let's say that you're going to estimate a mean and a covariance of something. We have a constraint here. There might be some constraints. Here's a constraint.

The covariance matrix has to be positive semi definite. That's a constraint, because if you created a covariance matrix that wasn't. That's a nonnegotiable constraint

Example 3

Data fitting
- variables: model parameters
- constraints: prior information, parameter limits
- objective: measure of misfit or prediction error

Negative log-likelihood is a loss function used in multi-class classification. Calculated as −log(y), where y is a prediction corresponding to the true label

Note: diffusion coefficient is a proportionality constant between the molar flux due to molecular diffusion and the gradient in the concentration of the species (or the driving force for diffusion) (or from many substances)

contd…

You could also have things like this. You could say these x's must be between this limit and that limit. For example, suppose you're estimating some diffusion coefficient or some parameters known to be positive. Then you should make it positive. These are parameter limits. The objective is dependent on how you want to describe the problem. For example, if I choose a bunch of parameters, I then propagate it through forward model and find out what I would have had, had this been the real parameter.

The question is, does it fit the observations well? Another way to say it – it's a measure of implausibility. That's really what it is. In this case, we're minimizing it. In many contexts, you'll see it turned around so it's maximized. You'd minimize the negative log likelihood function, which I think they call loss in some of these things.
Implausibility in a framework, a measure of implausibility would be something like the negative log posterior probability. If you minimize that, you're doing MAP, which is maximum a posteriori probability estimation.

Solving optimization problems

General optimization problem

- very difficult to solve
- methods involve some compromise, e.g., very long computation time, or not always finding the solution

Exceptions: certain problem classes can be solved efficiently and reliably

- least-squares problems
- linear programming problems
- convex optimization problems

This is related to the following – solving optimization problems. How do you solve them? It turns out it's really hard, and basically in general, I think it's fair to say it can't be done.

Methods to solve a general optimization problem – they always involve a compromise. The two main categories of compromise are as follows. The first one is to not back down on the meaning of solve. Solve means solve. It means find the point that has least objective and satisfies the constraints. That's the definition of solve.

You leave that intact, but you leave open the possibility that the computation time might involve thousands of years or something like that.
People call that global optimization, and it's a big field. It is almost entirely based on convex optimization.

Solving optimization problems

General optimization problem

- very difficult to solve
- methods involve some compromise, e.g., very long computation time, or not always finding the solution

Exceptions: certain problem classes can be solved efficiently and reliably

- least-squares problems
- linear programming problems
- convex optimization problems

There are cases where you can solve these problems. The most famous one in the world by far is least squares, least norm ($\|\mathbf{Ax} - \mathbf{b}\|_2^2$)

I transpose just x is equal to A transpose A inverse A transpose B. That often works super well in practice.

The status of that is that is the global solution. There are a couple others that you might not know about, and that would be things like linear programming problems.

The parent of these, is this convex optimization problem. The rough idea, to put it all together, is something like this.

Most optimization problems: A – everything is an optimization problem. B – we can't really solve most optimization problems.

When you do convex optimization, You solve the problem. It's the global solution.

Least-squares

Minimize $\|Ax - b\|_2^2$

Solving least-squares problems

- analytical solution: $x^\star = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- computation time proportional to n^2k ($A \in R^{k \times n}$); less if structured
- a mature technology

Using least-squares

- least-squares problems are easy to recognize
- a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

We have least squares or, if you're in statistics, regression. It may have other names. Here's the problem. You want to choose X to minimize the norm squared of Ax minus b. If A is full rank or skinny, you get an analytical solution, which you know if you know about linear algebra.

It's just A transpose A inverse A transpose B. In this case, it's a unique solution. In fact, we have a formula for it, which is overrated. There will be analytical formulas to almost none of them.

You'll have to wean yourself away from analytical formulas.

An optimization problem – you do not have to have any constraints, in which case it's called unconstrained, and you don't even have to have an objective. It's minimized. The universal solution – X equals zero.

Much more important than the formula – it turns out you can write down a formula for a lot of stuff, and it doesn't do you any good if you want to calculate it.

Here, there are reliable and efficient algorithms that will carry out this computation for you

Least-squares

Minimize $\|Ax - b\|_2^2$

Solving least-squares problems

- analytical solution: $x^\star = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- computation time proportional to n^2k ($A \in R^{k \times n}$); less if structured
- a mature technology

Using least-squares

- least-squares problems are easy to recognize
- a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

Here, just so you know, the computation time to solve a least squares problem goes n squared k. n is the number of variables, and that's the small dimension. k is the number of rows in A. It's a good thing to know. It's the small squared times the big.

Then, we did a couple thousand row least squares problem. You can call it a regression problem in 2,000 dimensions (variables) with 1,000 regressors (terms).
By the way, if A is sparse or has special structure – suppose part of A has an FFT embedded in it. That would come up in medical imaging.
You can do that faster. In image processing, you'd have transformations.
You can solve least squares that are way bigger.

I would say that least squares is a mature technology. I said by the way, I mean technology here. What it means is that other people know enough about the theory, the algorithms, and the implementations are so good

Least-squares

Minimize $\|Ax - b\|_2^2$

Solving least-squares problems

- analytical solution: $x^\star = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- computation time proportional to n^2k ($A \in R^{k \times n}$); less if structured
- a mature technology

Using least-squares

- least-squares problems are easy to recognize
- a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

If you learn a few tricks in least squares, you can do well. If you learn how to do weight twiddling and you learn about regularization, those two alone – you are now trained and ready to do battle with using least squares as a tool.

Weights is basically you go into a problem, and someone says there's no limit on the sum of the squares.
$\sum_{i=1}^{k} w_i \, (a_i^T x - b_i)^2$ where $w_1, \ldots, w_k$ are positive, is minimized

I have these weights here. You look at the least squares solution. You don't like what you see. You change the weights and you do it again.

Linear programming

minimize $c^T x$
subject to $a_i^T x \leq b_i$, i = 1,...,m

Solving linear programs
- no analytical formula for solution
- reliable and efficient algorithms and software
- computation time proportional to n^2m if $m \geq n$; less with structure
- a mature technology

Using linear programming
- not as easy to recognize as least-squares problems
- a few standard tricks used to convert problems into linear programs

(e.g., problems involving l_1- or l_∞-norms, piecewise-linear functions)

Linear programming is a problem that looks like this. Minimize a linear function subject to a bunch of linear inequalities.

Talk about the attributes of linear programming. The first is in general, except for very special cases, there's no analytical formula for the solution. There is no A transpose A inverse A transpose B.

There are reliable and efficient algorithms and software. you will write something 50 lines of Mat-Lab or Python, that will solve huge linear programs quite well- you get the solution. The computation is proportional to M squared N.

That's exactly the same as least squares. If you equate rows of the least squares objective with constraints, it's identical. M is a number of constraints here, and k is the height of A. So, this m – there's still that many rows in A.

Linear programming

minimize $c^T x$
subject to $a_i^T x \leq b_i$, i = 1,...,m

Solving linear programs
- no analytical formula for solution
- reliable and efficient algorithms and software
- computation time proportional to n^2m if $m \geq n$; less with structure
- a mature technology

Using linear programming
- not as easy to recognize as least-squares problems
- a few standard tricks used to convert problems into linear programs

(e.g., problems involving l_1- or l_∞-norms, piecewise-linear functions)

if I write that as a matrix inequality, Ax less than b, yes, that would be – this m would be that k. x is an R^n, so c is an R^n, too.

The modern era traces back to Stanford and George Dantzig. LP was something that you just talked about until you had computers, at which point LP looked a lot more interesting.

That was 1948. A lot of people knew about linear programming. Something like this coupled with computers – that's a mature technology.

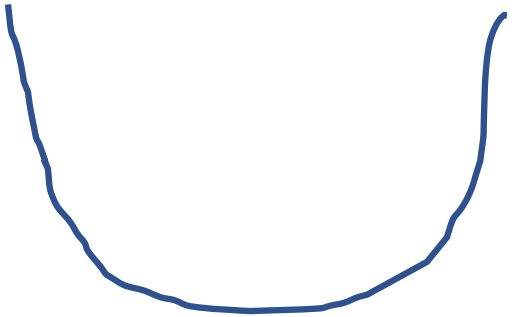
Convex optimization problem

minimize $f_0(x)$
subject to $f_i(x) \leq b_i$, i = 1,...,m

- Objective and constraint functions

are convex:
$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$
if $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$

- includes least-squares problems and linear programs as special cases



Graph looks like that. That's a convex function. The graph should have positive curvature c

Here's what it is. It's an optimization problem – minimize an objective subject in constraints. The function f_0 and the f_i 's have to be convex. You know what convex is. It means that the graph looks like that. That's a convex function. The graph should have positive curvature.

Least squares problem has that form because if I look at the least squares objective and I look at the plot of it, it's a quadratic function squared, and basically, it looks like a bowl. If you take a slice at a level set, you get an ellipsoid. It's convex. Linear programming also is a convex problem because all of the objectives are linear. Linear functions are convex. Linear functions are right on the boundary. They have zero curvature.

One way to say convex is just positive curvature. This includes least squares, and kind of the central idea at the highest level of this class is this. If you want to solve a convex optimization problem, there are no analytical solutions.
There are in special cases. We'll see tons of cases in communications and various other places where they have special analytical solutions. You've seen one in least squares already.

Convex optimization problem

Solving convex optimization problems

- no analytical solution
- reliable and efficient algorithms
- computation time (roughly) proportional to $\max\{n^3, n^2m, F\}$, where F is cost of evaluating f_i's and their first and second derivatives
- almost a technology

Using convex optimization

- often difficult to recognize
- many tricks for transforming problems into convex form
- surprisingly many problems can be solved via convex optimization

In general, there isn't an analytical solution. However, there are going to be reliable and efficient algorithms for solving these. You will get the solution. In fact, it would be very difficult to make an argument that solving a convex optimization problem compared to a least squares problem was, for example, that you'd been reduced to a numerical solution, which is what a lot of people might say with a hint of. They say numerical method in a way that makes you want to go wash your hands.

The computation time for solving convex problems is roughly proportional to something like n cubed – the number of variables (n) cubed – n squared m and F, where F is the cost of evaluating the functions ($f=\{f_1, .., f_m\}$ and their first and second derivatives. We don't have to get into that, but the point is it's like least squares. You can basically solve these.
You'll know how to write the algorithms to solve them and stuff like that. It is an exaggeration, in fact, to say it's a technology. It's almost a technology.

Convex optimization problem

Solving convex optimization problems

- no analytical solution
- reliable and efficient algorithms
- computation time (roughly) proportional to $\max\{n^3, n^2m, F\}$, where F is cost of evaluating f_i's and their first and second derivatives
- almost a technology

Using convex optimization

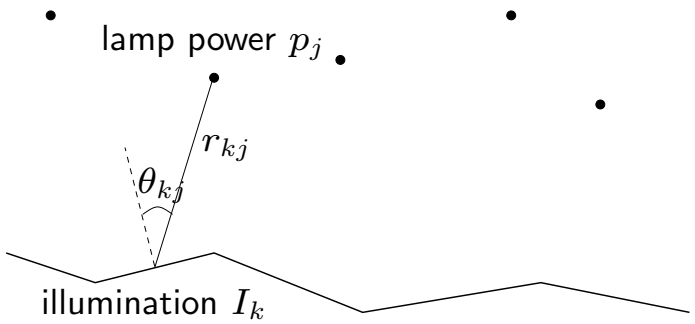
- often difficult to recognize
- many tricks for transforming problems into convex form
- surprisingly many problems can be solved via convex optimization

Optimization problems where the objectives and constraints have positive curvature, roughly speaking, we can solve. The theoretical Implication is extremely interesting. The practical ramifications of this are immense. It means you're messing on some problem in your own field, and if you get it to a problem of this form – it won't be obvious.

It's much cooler when you get to it and you turn things around and you switch some variables around. All the smoke clears. There are some functions there that it's not remotely obvious they're convex. You show they are. It means you can now solve those problems.

Example

m lamps illuminating n (small, flat) patches



intensity I_k at patch k depends linearly on lamp powers p_j:

$I_k = \sum_{j=1}^{m} a_{kj} p_j$; $a_{kj} = r_{kj}^{-2}$ max {cos θ_{kj}, 0}

Problem: achieve desired illumination I_{des} with bounded lamp powers

minimize $\max_{k=1,\ldots,n} |\log I_k - \log I_{des}|$

subject to $0 \le p_j \le p_{max}$, j = 1,...,m

Here's our problem. We have some lamps here. We're going to choose the illuminating powers on the lamps. That's going to be our variable.

The illumination on a patch here is the sum of the illumination from all the lamps here. It's going to be proportional to the lamp power and then the proportionality constant is going to be an inverse square law. r is the distance between the patch and the lamp.

The problem is to achieve a desired illumination which is given. You want to do this with bounded lamp powers. We have a maximum power. Powers cannot be negative. We care on a log scale, because what we care about is missing the lamp power by two percent or twelve percent. We don't care about absolute. It's ratio compared to this one. The question is how would we solve it?

Example

How to solve?

1.use uniform power: $p_j = p$, vary p

2.use least-squares:

minimize $\sum_{k=1}^{n} (I_k - I_{des})^2$

round p_j if $p_j > p_{max}$ or $p_j < 0$

3. use weighted least-squares:

minimize $\sum_{k=1}^{n} (I_k - I_{des})^2 + \sum_{j=1}^{n} w_j(p_j - p_{max}/2)^2$

iteratively adjust weights w_j until $0 \le p_j \le p_{max}$

4. use linear programming:

minimize $\max_{k=1,\ldots,n} |I_k - I_{des}|$

subject to $0 \le p_j \le p_{max}$, $j = 1,\ldots,m$

which can be solved via linear programming

of course these are approximate (suboptimal) 'solutions'

The first thing you do is you say let's try some simple suboptimal schemes. One is just this – you set all the lamps at the same power. You vary it and you plot this objective. You do a search over it. You do that. That might work, but it might not. That's a good baseline design.
You could say, well, we just learned least squares. You're going to use least squares.

The objective here is not the sum of the squares. Everyone uses the absolute value – the sums of the absolute values of the law of percentage error. We use the sum of the squares. When you solve this problem, I guarantee you some of the p's are going to come out negative.

What will happen is you'll look at the powers and a whole bunch of them will be negative. Then the heuristic. What do you do? Here's what you do. You say if a p is bigger than the maximum power, I just set it equal to p_{max}. If it's a negative lamp, I turn that lamp off

Example

How to solve?

1.use uniform power: $p_j = p$, vary p

2.use least-squares:

minimize $\sum_{k=1}^{n} (I_k - I_{des})^2$

round p_j if $p_j > p_{max}$ or $p_j < 0$

3. use weighted least-squares:

minimize $\sum_{k=1}^{n} (I_k - I_{des})^2 + \sum_{j=1}^{n} w_j(p_j - p_{max}/2)^2$

iteratively adjust weights w_j until $0 \le p_j \le p_{max}$

4. use linear programming:

minimize $\max_{k=1,\ldots,n} |I_k - I_{des}|$

subject to $0 \le p_j \le p_{max}$, $j = 1,\ldots,m$

which can be solved via linear programming

of course these are approximate (suboptimal) 'solutions'

Once again, you see how well you did, and you might do better than uniform – maybe worse. Now, this is what someone trained in least squares would do. They'd say not a problem. They'd go over here and say I want p_j to be in the interval zero p_{max}. Therefore, I'm going to add a regularization term which charges p for being away from the center of that interval.

You start with all the w's one, and you solve this problem. Then, you just start weight twiddling. You'll come up with some algorithm that twiddles the weights, and how you twiddle them is totally obvious. If p comes out outside that interval, that weight needs to go up in the next cycle. If p is timid because your weight is so high, you want to decrease that weight. A couple of cycles of this, and you're going to get a good solution.

You could also use linear programming. If you know about linear programming, you could actually solve this L one problem where you minimize an L one norm. This is closer than that. Linear programming – it handles the inequalities. This would probably do the best of all of them.

Example

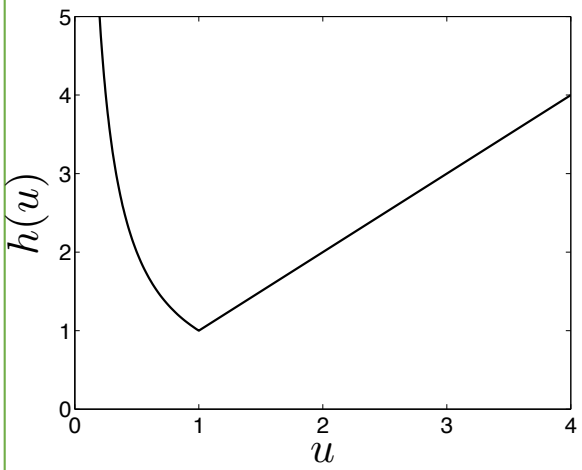
How to solve?

5. use convex optimization: problem is equivalent to

minimize $f_0(p) = \max_{k=1,\ldots,n} h(I_k/I_{des})$

Subject to $0 \le p_j \le p_{max}$, $j = 1,\ldots,m$

with $h(u) = \max\{u, 1/u\}$



f_0 is convex because maximum of convex functions is convex

exact solution obtained with effort ≃ modest factor × least-squares effort

The problem is convex, so this objective function – it just looks like this. It's linear over here, and then it's an inverse over here. You look at that function and you realize a couple of things. The important part is looking for convexity. This is what we like to see.

By the way, you will see that that function is not differentiable. In a lot of other treatments of optimization, differentiability has this very high role, because a lot of things are based on gradient and derivatives, and there's no derivative there. So in convex optimization, differentiability is going to play a much lower role.

This problem, even though it is non-differentiable here, it can be solved as fast as least squares if you know what you're doing. We might even have you write from scratch a solver for it. This is just an example of a problem where it's not obvious exactly how to solve this.

Nonlinear optimization

Traditional techniques for general nonconvex problems involve compromises

Local optimization methods (nonlinear programming)
- find a point that minimizes f_0 among feasible points near it
- fast, can handle large problems
- require initial guess
- provide no information about distance to (global) optimum

Global optimization methods

- find the (global) solution
- worst-case complexity grows exponentially with problem size

these algorithms are often based on solving convex subproblems

Brief history of convex optimization

Theory (convex analysis): ca1900–1970

Algorithms

- 1947: simplex algorithm for linear programming (Dantzig)
- 1960s: early interior-point methods (Fiacco & McCormick, Dikin, . . .)
- 1970s: ellipsoid method and other subgradient methods
- 1980s: polynomial-time interior-point methods for linear programming (Karmarkar 1984)
- late 1980s–now: polynomial-time interior-point methods for nonlinear convex optimization (Nesterov & Nemirovski 1994)

Applications

- before 1990: mostly in operations research; few in engineering
- since 1990: many new applications in engineering (control, signal processing, communications, circuit design, . . .); new problem classes (semidefinite and second-order cone programming, robust optimization)