Analysis of Mathematical Optimization

Some Classes, Problems, and Algorithms

by Liam Wrubleski

April 2020

Contents

Overview of Mathematical Optimization

General Overview

- The general idea of mathematical optimization is to find an item x in a set D that minimizes the value of a function $f : D \to \mathbb{R}$.

General Statement

Find

$$m \in \mathbb{R}, x_0 \in D$$

such that

$$m = f(x_0) = \min_{x \in D}\{f(x)\}$$

for some set D, and

$$f : D \to \mathbb{R}$$

- The function f is referred to as the **objective function**.
- In general, the set D may be any set, but here we only analyze sets $D \subseteq \mathbb{R}^n$, for any positive integer n (this is called **continuous optimization**).
- Often, the set D is specified using **constraint functions**:

$$D = \{\mathbf{x} \in \mathbb{R}^n | g_i(\mathbf{x}) \leq 0, h_j(\mathbf{x}) = 0\}$$
$$g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, ..., p$$
$$h_j : \mathbb{R}^n \to \mathbb{R}, j = 1, ..., q$$

- The functions g_i are **inequality constraints**, and h_j are **equality constraints**.

- In this case, we consider $f : \mathbb{R}^n \to \mathbb{R}$, as exploring potential solutions outside of D allows for different solving methods.
- When the problem is of the above form, it is generally expressed in **standard form**

Standard form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, p \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, q \end{aligned}$$

Equality & Inequality

Any equality constraint may be expressed using two inequality constraints, using

$$h_j(\mathbf{x}) \leq 0, -h_j(\mathbf{x}) \leq 0$$

Some optimization classes use only inequality constraints, but some take advantage of equality constraints to simplify computation.

Maximization & Minimization

Problems attempting to maximize the value of $f(\mathbf{x})$ with some constraints can be computed by minimizing the value of $-f(\mathbf{x})$ subject to the same constraints, so maximize and minimize are effectively interchangeable.

Linear Programming - LP

Terminology

Programming

Programming does not refer to computer programs. It was a term used by the US military to refer to proposed schedules, and made its way into mathematics via George B. Dantzig.

Linear

Linear in this context generally refers to affine in general mathematics. Linear programs are not themselves linear, but make use of the affine properties of their description.

- Linear Programming (LP) is a very simple class of mathematical optimization using only inequality constraints
- When expressed in standard form, a linear programming problem has the following properties

$$
\begin{aligned}
f(\mathbf{x}) &= c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\
g_i(\mathbf{x}) &= a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{in} x_n - b_i, i = 1, \ldots, m \\
x_j &\geq 0
\end{aligned}
$$

- This is usually expressed in standard form as

$$
\begin{aligned}
\text{maximize} \quad & \mathbf{c}^T \mathbf{x} \\
\text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

- The objective function is linear, but the constraints are not. They are **affine**.

Affine Functions

We call a function $f : \mathbb{R}^n \to \mathbb{R}$ affine if and only if

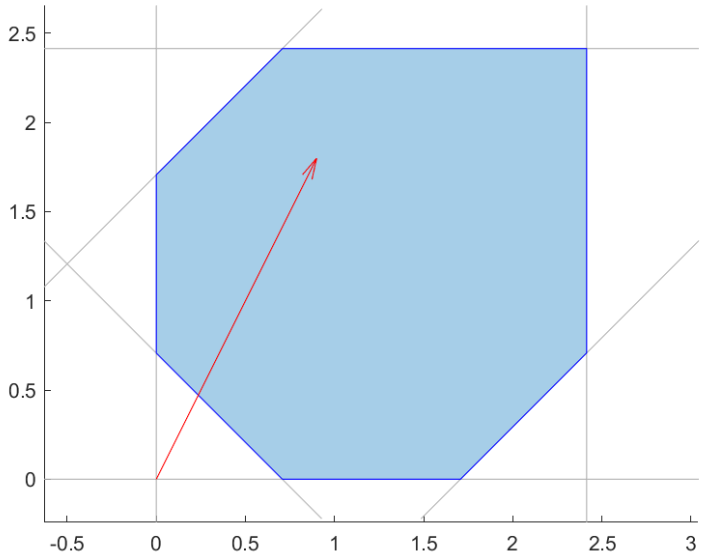
$$f(\mathbf{x}_1) = f(\mathbf{x}_2) = a \implies f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) = a, \quad \forall \alpha \in \mathbb{R}, \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$$

- Affine functions are more general than linear functions.

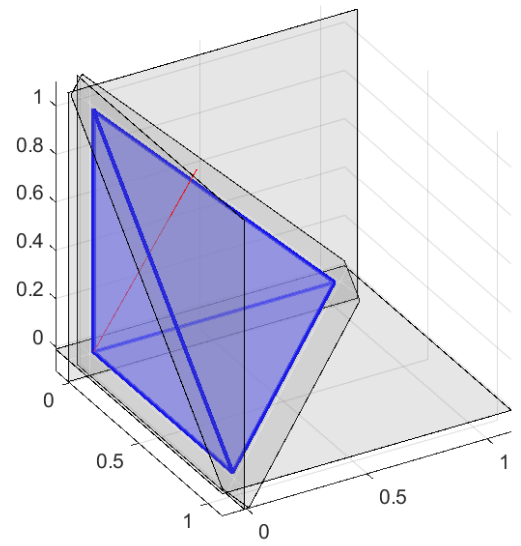
- The space of possible solutions satisfying all of the constraints may be considered an n-polytope.
- For $n = 2, n = 3$, this gives us an intuitive geometric interpretation, shown for two contrived examples in 2d and 3d respectively on the following slides.
- The blue polytope represents the space of possible solutions, the grey lines/planes represent the constraints, and the red arrow represents the direction of optimization.

Linear Programming - LP

Theory of Linear Programming



$$
\begin{aligned}
\text{maximize} \quad & x + 2y \\
\text{subject to} \quad & -x - y + \frac{1}{\sqrt{2}} \leq 0 \\
& -x + y - \frac{\sqrt{2}+1}{\sqrt{2}} \leq 0 \\
& -x + y + \frac{\sqrt{2}+1}{\sqrt{2}} \leq 0 \\
& x - 1 - \sqrt{2} \leq 0 \\
& y - 1 - \sqrt{2} \leq 0 \\
& x, y \geq 0
\end{aligned}
$$



$$
\begin{aligned}
\text{maximize} \quad & x + y + 3z \\
\text{subject to} \quad & x + y + z - 1 \leq 0 \\
& x, y, z \geq 0
\end{aligned}
$$

Linear Programming - LP

Simplex Algorithm

- Developed by George B. Dantzig in 1947 to solve LP problems, using the LP formulation of Leonid Kantorovich.
- Finds an optimal solution to an LP problem by moving along the edges of the polytope associated with the problem.
- Efficient on random problems, although inefficient in the worst case.

- For each inequality $g_i(\mathbf{x}) \leq 0$, a slack variable $s_i \geq 0$ is added to make an equality of the form $g_i(\mathbf{x}) + s_i = 0$.
- The problem is then arranged into a matrix referred to as a **tableau**:

$$\begin{bmatrix} 1 & \mathbf{c}^T & \mathbf{0}_m^T & 0 \\ \mathbf{0}_m & A & I_m & \mathbf{b} \end{bmatrix}$$

  where I_m is the $m \times m$ identity matrix
- We then move along the edges of the polytope for the original problem by performing operations called **pivots** on this tableau.

Linear Programming - LP
Simplex Algorithm

- The exact operation of the algorithm takes too long to examine here, but will be examined in my paper.
- However, as a high level examination, each pivot moves along one edge of the polytope connected to the current point.
- This requires a number of row operations equal to the number of constraints in the model.
- We will examine the intermediate results of the algorithm for a small example.

Linear Programming - LP

Simplex Example

- Suppose a farmer has 10 km^2 of land on which to grow wheat and barley, and that she has 17 kg of fertilizer and 13 kg of pesticide to use to grow it.
- Every 1 km^2 of wheat requires 2 kg of fertilizer and 2 kg of pesticide, and sells for \$6000 of profit.
- Every 1 km^2 of barley requires 3 kg of fertilizer and 1 kg of pesticide, and sells for \$5000 of profit.
- Suppose that the farmer needs to plant at least 1 km^2 of crops, regardless of any other consideration.

How should the farmer plant her crops to maximize her profit?

Linear Programming - LP

Simplex Example

- Let x be the amount of land used for wheat, and y be the amount of land used for barley.
- Total 10 km^2 of land $\implies x + y \leq 10$.
- Total 17 kg of fertilizer $\implies 2x + 3y \leq 17$.
- Total 13 kg of pesticide $\implies 2x + y \leq 13$.
- Plant at least 1 km^2 of land $\implies x + y \geq 1$.
- Maximize profit (in thousands of dollars)
  $\implies \text{maximize } 6x + 5y$.

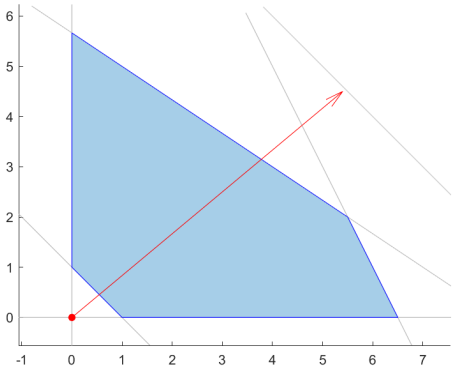
Linear Programming - LP

Simplex Example

$$
\begin{array}{ll}
\text{maximize} & 6x + 5y \\
\text{subject to} & x + y - 10 \leq 0 \\
& 2x + 3y - 17 \leq 0 \\
& 2x + y - 13 \leq 0 \\
& -x - y + 1 \leq 0 \\
& x, y \geq 0
\end{array}
$$

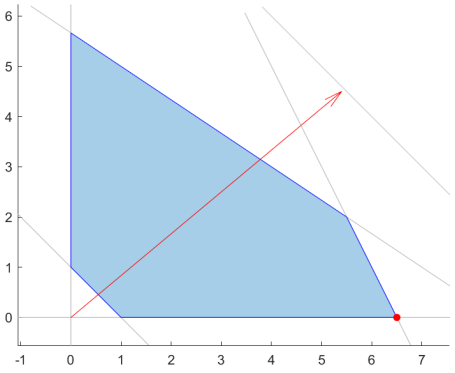
Linear Programming - LP

Simplex Example

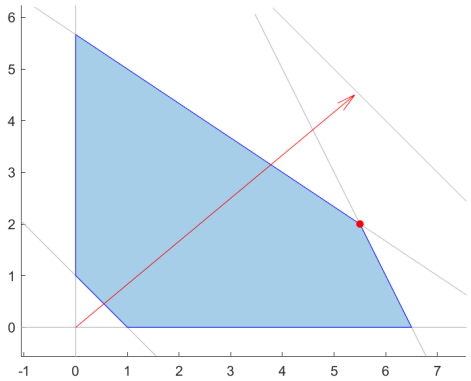


$$x = 0$$
$$y = 0$$

Note this point is not actually in the feasible space!



$$x = 6.5$$
$$y = 0$$



$$x = 5.5$$
$$y = 2$$

This is our optimal point, with a maximized profit of \$43,000.

Linear Programming - LP

Analysis

- Each pivot requires $m - 1$ row operations, and each row operation requires $n + m + 2$ additions and $n + m + 2$ multiplications.
- When the algorithm is written with a good selection criterion for which edge to go along, it will not revisit any vertex of the polytope. However, as shown by Klee and Minty, there exist problems for which the algorithm will visit each vertex before terminating.
- This gives us that its worst case complexity is $O((nm + m^2)v)$, where v is the number of vertices. However, as shown by Daniel Spielman, the average number of vertices visited is polynomial in the number of dimensions.

Klee-Minty Cube

Klee and Minty showed that on the following problem, the usual simplex algorithm implementation visits every vertex of the polytope, giving a worst case complexity of $O(2^n)$.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{D} 2^{D-i} x_i \\ \text{subject to} \quad & \sum_{i=1}^{k-1} 2^{D-i+1} x_i + x_k + k \leq 5^k, k = 1, ..., D \\ & x_i \geq 0, i = 1, ..., D \end{aligned}$$

The polytope for this problem looks like a squashed hypercube, which is where the name Klee-Minty Cube originates.

Mixed Integer Linear Programming - MILP

Theory

- A variant of linear programming, with the additional requirement that some or all of the variables must have integer values.
- This is a significantly more difficult variation of the problem.

How much harder is it?

For reference, with just 120 variables, each of which can be either 0 or 1, computing all possibilities would take 16 times longer than the age of the universe on the world's fastest supercomputer, assuming each possibility takes only one floating point operation. Allowing each variable to also be 2, it would take a billion trillion times the age of the universe.

Mixed Integer Linear Programming - MILP

Theory

- The numbers in the previous slide may seem unreasonable, but many of the best known methods for solving these problems devolve to brute-force methodology.
- Further, although the examples up to now use very few variables, real-world problems often have hundreds of variables. For example, the standard method of assigning 40 people to 40 tasks uses at least 1600 integer variables.

Mixed Integer Linear Programming - MILP

Theory

- The way these problems are solved begins by solving the LP-relaxation of the problem.

LP Relaxation

The LP-Relaxation of the MILP problem

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, i = 1, ..., m \\ & x_i \geq 0, i = 1, ..., n \\ & x_i \in \mathbb{Z}, i = 1, ..., k \leq n \end{aligned}$$

is an LP problem with the same objective and inequality constraints, but without the integrality constraints.

Mixed Integer Linear Programming - MILP

Theory

- Once the LP relaxation has been solved, it is used to find another solution to the constraints, but hopefully closer to a solution satisfying the integrality constraints. This is then repeated until it has been shown that the most recently found integer solution is optimal.
- The first common way of doing this is called **branch and bound**.
- The second common way of doing this is called **cutting planes**.

Mixed Integer Linear Programming - MILP

Branch and Bound

- The value of the objective function for the solution to the LP relaxation is an upper bound for the value of the objective function for the MILP.
- If any integral solution is found, the value of the objective function for that solution is a lower bound for the value of the objective function for the MILP.
- If any integral solution is found with the value of the objective function equal to the upper bound, that solution is optimal.

Mixed Integer Linear Programming - MILP

Branch and Bound

- Suppose the solution to the LP relaxation has some variables which are non-integer, but which have integrality constraints in the original MILP. Take x_i to be one of these variables with value x_i'(the best methods for choosing which variable are generally trade-secrets, but many methods can be used).
- Create two new LP problems, one with the additional constraint $x_i \leq \lfloor x_i' \rfloor$, and the other with $x_i \geq \lceil x_i' \rceil$.
- Solve each of these problems using branch and bound. If at any point the solution to the LP relaxation is worse than the current lower bound, prune that branch. If at any point the solution to the LP relaxation is integral, and is better than the current lower bound, update the lower bound.

Mixed Integer Linear Programming - MILP

Cutting Planes

- Given any non-integral solution, a cutting plane can be generated that removes the current solution, but no potential integral solutions.
- While this is unlikely to produce a solution by itself, it decreases the upper bound of the MILP and potentially produces a better configuration for branching.

Mixed Integer Linear Programming - MILP

Branch and Cut

- This method combines branch-and-bound methods and cutting-plane methods to more quickly approach an integer value.
- Unfortunately, the best methods for balancing these techniques is an industry secret, and largely heuristic.
- In the worst case, branch-and-bound degenerates to brute force, and cutting planes generally do not help find integer solutions quickly, so even the best techniques are very difficult.

Mixed Integer Linear Programming - MILP

Interesting Problems

- Several well-known difficult problems can be expressed as MILP problems
- The knapsack problem is very easily converted to a MILP.
- The boolean satisfiability problem is also easily converted to a MILP.
- Many other NP-complete problems are also easily converted to MILP problems.
- Real-world scheduling problems have a straightforward conversion to MILP problems.

Mixed Integer Linear Programming - MILP

Problem Conversion

Knapsack Problem

Consider n items, each of which has a value v_i, a weight w_i, and a size s_i. You have a knapsack with a maximum weight W and a maximum volume S. The knapsack problem is to determine which items should be placed in the knapsack to maximize the carried value. This can be expressed as a MILP as follows:

$$\begin{aligned} \text{maximize} \quad & \mathbf{v}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{w}^T\mathbf{x} \leq W \\ & \mathbf{s}^T\mathbf{x} \leq S \\ & x_i \in \{0,1\}, i = 1, ..., n \end{aligned}$$

Here, x_i is 1 if you should pack item i, and 0 otherwise.

Mixed Integer Linear Programming - MILP
Problem Conversion

- The type of variable used in the above conversion is a **decision variable**. It's binary, and corresponds to making a decision (in this case, pack the item or not).
- Not all integer constraints are decision variables. For example, if we had an arbitrary amount of each item in the above problem, we could have the variables correspond to how many of each item we pack.
- These variables would still have an integrality constraint, as we generally assume we cannot pack part of an item.

Convex Optimization
Theory

- Linear programming is one of the simplest classes of convex optimization, and while it isn't convex, mixed-integer linear programming is often studied in the same areas.
- There are a lot of different kinds of convex optimization, but they all share some characteristics
  - The objective function is convex, and
  - The constraints are convex.

Convex Optimization
Theory

Convex Sets

A set S is convex iff for any two points a, b, and any $\alpha \in [0, 1]$, $\alpha a + (1 - \alpha)b$ is also in the set S. Intuitively, if a line segment is drawn between the points a and b, the line lies entirely within the set S.

Epigraph

The epigraph of the function $f : \mathbb{R}^n \to \mathbb{R}$ is a set $S \subseteq \mathbb{R}^{n+1}$, with

$$S = \{(x_0, x_1, ...x_n) | x_0 \geq f(x_1, ...x_n)\}$$

Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff its epigraph is convex.

Convex Optimization
Theory

- Convex optimization is simple relative to non-convex optimization, as if a local minimum of the objective function is found in the interior of the feasible space, then it is a global minimum.
- If a local minimum is not found in the interior of the feasible space, then the minimum of the objective function on the feasible space is in the boundary of the feasible space (which is why strict inequalities are not used in LP).

Note

In LP and MILP, the sense is typically maximization. In convex optimization, the sense is typically minimization.

Convex Optimization
Quadratic Programming - QP

- Quadratic programming allows the objective function, but not the constraints functions, to take the form

$$f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

- One special case of quadratic programming is ordinary least squares, which will be explained shortly.
- Note that Q must be positive semidefinite for f to be convex.

Ordinary Least Squares

Given N test cases $\mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}$, find the constant vector $\mathbf{k} \in \mathbb{R}^n$ that minimizes the sum of the squared errors

$$
\begin{aligned}
E &= \sum_{i=1}^{N} (\mathbf{k}^T \mathbf{x}_i - y_i)^2 = \sum_{i=1}^{N} (\mathbf{k}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{k} - 2 y_i \mathbf{x}_i^T \mathbf{k} + y_i^2) \\
&= \mathbf{k}^T (\sum_{i=1}^{N} \mathbf{x}_i^T \mathbf{x}) \mathbf{k} + (\sum_{i=1}^{N} -y_i (2\mathbf{x}_i)^T) \mathbf{k} + \sum_{i=1}^{N} y_i^2 \\
&= \mathbf{k}^T Q \mathbf{k} + \mathbf{c}^T \mathbf{k} + d
\end{aligned}
$$

As d is a constant, any solution minimizing E also minimizes $\mathbf{k}^T Q \mathbf{k} + \mathbf{c}^T \mathbf{k}$, and vice versa. This shows that ordinary least squares is an unconstrained quadratic programming problem.

Convex Optimization
Quadratically Constrained Quadratic Programming - QCQP

- A generalization of quadratic programming that allows the constraint functions to also take the form of convex quadratic functions, so the standard form looks like

$$\begin{aligned} \text{minimize} \quad & \mathbf{x}^T Q_0 \mathbf{x} + \mathbf{p}_0^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{x}^T Q_i \mathbf{x} + \mathbf{p}_i^T \mathbf{x} + r_i \leq 0, \quad i = 1, ..., m \\ & A\mathbf{x} = \mathbf{b} \end{aligned}$$

  where all $Q_i, i = 0, ..., m$ are positive semidefinite.
- Problems that can be formatted as QCQP problems come up often in statistics (for example, constraining the variance of a random variable)

Portfolio optimization

Suppose you are trying to invest in n different stocks, represented by the random variable $\mathbf{p} \in \mathbb{R}^n$ with known mean $\bar{\mathbf{p}}$ and covariance Σ, where p_i represents the price of stock i at some point in the future divided by its price now. Let x_i denote the amount you invest in stock i, and B your total budget. Then your return is the random variable $\mathbf{p}^T\mathbf{x}$ with mean $\bar{\mathbf{p}}^T\mathbf{x}$ and variance $\mathbf{x}^T\Sigma\mathbf{x}$. Using the variance as a metric of risk, you might come up with this:

$$\begin{array}{ll} \text{minimize} & -\bar{\mathbf{p}}^T\mathbf{x} \\ \text{subject to} & \mathbf{x}^T\Sigma\mathbf{x} - \sigma_{max} \leq 0 \\ & \mathbf{1}^T\mathbf{x} = B \end{array}$$

which maximizes your return, subject to a maximum allowable risk.

Convex Optimization
Second-Order Cone Programming - SOCP

- A generalization of QCQP, this effectively allows the matrix Q to have a single negative eigenvalue (making it no longer positive semidefinite. It's generally phrased, however, as follows

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}^T\mathbf{x} \\ \text{subject to} \quad & \|A_i\mathbf{x} + \mathbf{b}_i\|_2 \leq \mathbf{c}_i^T\mathbf{x} + d_i, \quad i = 1, \ldots, m \\ & F\mathbf{x} = \mathbf{g} \end{aligned}$$

- This kind of optimization is very useful in many real-world engineering problems, as the 2-norm can be used to constrain distances and placements of items.

- Geometric programs have a slightly different standard form to other optimization problems

$$\begin{array}{lll} \text{minimize} & f_0(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) \leq 1, & i = 1, ..., p \\ & g_i(\mathbf{x}) = 1, & i = 1, ..., q \end{array}$$

where $f_i(\mathbf{x})$ are **posynomials**, and $g_i(\mathbf{x})$ are **monomials**.

Monomials & Posynomials

The word monomial has a different meaning in geometric programming from the rest of mathematics. Here, it means a function from $\mathbb{R}^n_{++} \to \mathbb{R}$, defined as $x \mapsto c x_1^{a_1} x_2^{a_2} ... x_n^{a_n}$, with $c > 0, a_i \in \mathbb{R}$.
A posynomial is any sum of this type of monomials.

Convex Optimization
Geometric Programming - GP

- GP problems are not generally convex, but all GP problems can be transformed into an equivalent convex optimization problem. By taking $y_i = \log(x_i), \tilde{f}_i(\mathbf{y}) = \log(f_i(\exp \mathbf{y})), \tilde{g}_i(\mathbf{y}) = \log(g_i(\exp \mathbf{y}))$, we end up with a convex optimization problem in $\mathbf{y}$.
- Converting f_i this way results in $\tilde{f}_i$ being a weighted log-sum-exp function, which is convex. The proof of convexity is too long to show here, but is shown in the paper.
- Converting g_i this way results in $\tilde{g}_i$ being an affine function in y. We show this on the next slide.

Converting a monomial

The conversions made are a change of variables $y_i = log(x_i)$, and taking the logarithm of the function itself.

$$
\begin{aligned}
g(\mathbf{x}) &= c x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n} \\
\log(g(\mathbf{x})) &= \log(c) + a_1 \log(x_1) + a_2 \log(x_2) + \ldots + a_n \log(x_n) \\
\log(g(\exp(\mathbf{y}))) &= \log(c) + a_1 y_1 + a_2 y_2 + \ldots + a_n y_n \\
\tilde{g}(\mathbf{y}) &= \mathbf{a}^T \mathbf{y} + \log(c)
\end{aligned}
$$

- Overall, this gives that the conversion of a GP problem as expressed above is the convex problem

$$\begin{aligned} \text{minimize} \quad & \tilde{f}_0(\mathbf{y}) \\ \text{subject to} \quad & \tilde{f}_i(\mathbf{y}) \leq 0, \quad i = 1, ..., p \\ & \tilde{g}_i(\mathbf{y}) = 0, \quad i = 1, ..., q \end{aligned}$$

  where $\tilde{f}_i(\mathbf{y})$ are weighted log-sum-exp functions, and $g_i(\mathbf{x})$ are affine functions, all of which are convex, so this problem is convex.

- Geometric programming often arises in electrical engineering, regarding the size of components, as well as in statistics for maximum likelihood estimation.

- This is the most general class of convex optimization discussed here, and is also the hardest to solve. However, every other convex optimization class expressed above can also be expressed as an SDP problem.
- This means a solver for SDP problems can solve all of the problems listed above, so research into SDP solvers is more advanced than research into specific solvers for most of the above problems (with the exceptions of LP and MILP).

- The standard form for SDP problem is also slightly different from the other programs listed above, requiring the definition of the following inner product for A, B real nxn symmetric matrices:

$$\langle A, B \rangle = \sum_{i=1,j=1}^{n} A_{ij} B_{ij}$$

- Then the standard form of an SDP problem is

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \langle A_k, X \rangle \leq b_k, \quad k = 1, ..., m \\ & X \succeq 0 \end{aligned}$$

where C, A_k are symmetric nxn matrices, and $X \succeq 0$ means X is constrained to be positive semidefinite.

- As the objective function and inequality constraints are linear in X, it may not be obvious how this encompasses quadratic functions. This comes from the positive semidefiniteness constraint on X, but this is difficult to show here. It is gone into in more depth in the paper.
- SDP problems are usually solved with **interior point methods**, which trace a path through the interior of a space, as opposed to along its surface, as with the simplex algorithm.

Non-Convex Optimization

Theory

- The vast majority of potential optimization problems are not convex. Some of these problems have an equivalent convex problem, as with the GP problems shown above.
- However, some of these problems do not have equivalent convex problems.
- Non-convex problems may have several local extrema that are not globally optimal.
- Moreover, in real-world optimization problems, we may encounter objective functions or constraints that do not have a closed form.

Non-Convex Optimization
Theory

- In many engineering applications, we would like a computer to generate a solution to a problem (often a design fitting certain parameters).
- However, it is often the case that the performance of the potential solutions may only be evaluated through simulation
- Simulating part performance not only has no closed form, but is also extremely computationally expensive.
- It is desirable to find optimization algorithms that operate on black-box objective functions, and minimize the total number of evaluations of those objective functions.

Non-Convex Optimization

Wang et al. Stochastic Global Optimization

- In "Mode-pursuing sampling method for global optimization on expensive black-box functions", Wang et al. detail an optimization algorithm that finds the global optimum of an expensive black-box function (likely non-convex).
- This is a kind of **stochastic optimization**, which uses a stochastic process in its evaluation.
- Much of this paper details the methods used for generating the random points. This is interesting, but out of scope for this presentation, so the following slides do not detail the methods used to generate the random points. This is summarized in my paper.

Note

This algorithm assumes, without loss of generality, that f is positive on a compact set $S \subset \mathbb{R}^n$. This is valid, as either $\inf_{x \in S} f(x) = -\infty$, so f is unbounded, or $\inf_{x \in S} f(x) = a$, for some $a \in \mathbb{R}$. Then $f(x) + a + 1 > 0$, and any x minimizing $f(x) + a + 1$ also minimizes $f(x)$.

Non-Convex Optimization

Wang et al. Stochastic Global Optimization

- This algorithm has four primary steps:
  1. Generate a small number m of points $x^{(1)}, ...x^{(m)}$ using the current PDF g (initialized to be uniform over S).
  2. Evaluate the black box function at these points, giving $(x^{(i)}, f(x^{(i)})$. Use these points, along with any previous points evaluated, to generate an approximation $\hat{f}$ of f, such that $\hat{f}(x^{(i)}) = f(x^{(i)}) \forall i$.
  3. Take some $c \in \mathbb{R}$ such that $c > \hat{f}(x) \forall x \in S$, and let the PDF $g(x) = k(c - \hat{f}(x))$, where k is a normalizing constant so that $\int_S g(x) dS = 1$.
  4. If some stopping criterion is met, return the generated point with the best value. Otherwise, go to step 1.

- This algorithm works because the points generated in step 1 tend to cluster around the modes of g. These are the maxima of g, and by definition of g these points are the minima of $\hat{f}$. Furthermore, by keeping previously evaluated points, the approximation $\hat{f}$ of f gets more and more accurate.
- Because we have that $g(x) > 0 \forall x \in S$, every area in the feasible region has a non-zero probability of being sampled, which ensures that if the algorithm continued forever, it would be guaranteed to find the global minimum on S.
- With the stopping criterion, it is no longer guaranteed to find the global minimum, but it will find local minima, and select the best of these.

Non-Convex Optimization

Wang et al. Stochastic Global Optimization

- Wang et al. modify the basic algorithm above to detect nearly-quadratic regions of f, and with a speed control factor that adjusts how likely the algorithm is to select points away from the modes of g.
- These drastically speed up the algorithm, by adjusting its local vs global searching behaviour.

Non-Convex Optimization

Wang et al. Stochastic Global Optimization

This algorithm has several benefits.

- It attempts to minimize the number of objective function evaluations for a given problem.
- It supports parallel computation, as each evaluation of the objective function is independent of the others.
- It is also efficient on problems with inexpensive objective functions.

It does, however, occasionally get "stuck" in a local optimum, and in real-world circumstances the stopping criterion may prevent it from finding the global optimum.

Conclusion

Today, we went over several classes of optimization problems, explored the theory behind them, where they are useful, and some methods for solving them. I hope you found this interesting, and if you have any questions please feel free to ask me, and read my paper!

The End