Continuous Optimization and its Applications

Speaker: Prof. Henry Wolkowicz
Assistance by: Nathan Krislock

Contents

1 Introduction

(We had some time to kill to wait for the second bus, so...) Some commands in MatLAB that we will see: `optimtool`, `linprog`, `fminimax`. Check out the NEOS website for server-side optimization.

Have quadratic model q_k, a second-order Taylor series. Newton's Method, quadratic convergence, iterative procedure. This procedure is “scale free”.

Quadratic Q must be PSD on the feasible set.

Let Q be any orthogonal matrix. Consider an affine transformation

$$x \mapsto Qx + b.$$

NP-hard.

MOSEK software.

2 What is Optimization?

- Two quotes from Tjalling C. Koopmans, (1975 Nobel Memorial Lecture):
  - “best use of scarce resources”
  - “Mathematical Methods of Organizing and Planning of Production”
- Quote from Roger Fletcher (text, 1987): ”The subject of optimization is a fascinating blend of heuristics and rigour, of theory and experiment.”

We discuss some basic models of optimization problems.

Unconstrained optimization: The problem is

$$\text{Minimize } f(x)$$

where $x \in \mathbb{R}^n$ and and f is a continuous real valued function.

Linear programming: The basic problem of linear programming is to minimize a linear objective function of continuous real variables, subject to linear constraints. For purposes of describin g and analyzing algorithms, the problem is often stated in the standard form

$$\text{Minimize } c \cdot x \text{ subject to } Ax = b, x \geq 0.$$

Linear programs are solved using George Dantzig's simplex method or with interior point methods.

Quadratic programming:

Semidefinite (and Second Order) Cone programming: This is an area that's relatively new. Falls into the class of cone programming problems. If we think of x as being a vector, we have to change the vector space to the space of symmetric matrices. Now we draw a big X. The variable is now a matrix, and we are looking for a matrix that has to be PSD. The constraint $Ax = b$ in the linear form turns into $Aopp(x) = b$. The objective function in this case is to minimize $\langle C, X \rangle = trace(CX)$.

Originally, this was studied as an extension of linear programming, but the applications that come out are interesting. The linear transformation $AoppX$ is the same as a family of m equations $\langle A_k, X \rangle = b_k$.

Every linear program has a dual program. You can't do optimization without duality. Even unconstrained optimization, the derivative is in the dual space. Just like in LP, there's a dual.

We'll talk about this dual as we go along.

There's a class of problems called "second order cone problems". We have a cone

$$Q = \left\{ (x_0, \ldots, x_n) : x_0^2 \geq \sum_{i=1}^{n} x_i^2 \right\}$$

This cone is sometimes called an ice cream cone because of when $n = 3$.

One application is the "max cut" problem. Suppose we have a graph. Can give edges weights. We want to cut the graph into two, make two sets for the nodes, such that maximize the sum of the weights that are cut. There are applications of this in physics and VLSI design, but also mathematically interesting. We can model this with a quadratic objective.

Let x_i correspond to $+1$ or -1, depending on which set the vertex goes into. Thus, $x \in \{-1, +1\}^n$. Note, if x_i and x_j are in the same sign iff they are on the same side. So, in the sum below, we have zeroes or twos (which explains the denominator):

$$t := \max_x \frac{1}{2} \sum_{i<j} (1 - x_i x_j)$$

Now, we do something very silly. The feasible set here is the vertices of the n-cube, and they're disconnected, so it seems that calculus isn't going to help you. Don't the other points seem so far away? We'll replace the constraint by a quadratic $x_i^2 = 1$. It doesn't seem to do anything. But here's where we see Lagrange multpliers. Take the constraint and put it with a Langrange multiplier.

$$t(\lambda) := \max_x x^T Q x + \sum_i \lambda_i (1 - x_i^2)$$

(Here, the Laplacian Q is built off the quadratic.) But sometimes, we can't solve this. The Hessian looks like $x^T(Q - Diag(\lambda))x$, where $Diag(\lambda)$ is the $n \times n$ diagonal matrix

$$\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

The new Hessian is $Q - Diag(\lambda)$. The corresponding eigenvector gives you infinity. λ is a parameter, but each time we fix it, we get a t. If $\lambda = 0$, we get the original problem, but we allow other values. So this problem is a relaxation. Every time we solve this problem, we get a bound. It's a waste of time if an eigenvalue problem is positive. There's a hidden semidefinite constraint. If we're gonna get a good upperbound, throw out positive infinity. So, our constraint is $Q - Diag(\lambda) \leq 0$ (negative semidefinite).

The PSD cone is a nice convex set. The faces are well-described. Each face of a non-negative orthant corresponds to a cone in a smaller-dimensional cone.

The important thing is that $\lambda \leq t(\lambda)$. We can use Lagrange multiplier to relax. Of all the λ's, find the best one, so we minimize over λ. However, there's a hidden constraint. We can throw away the maximization $\max_x x^T Q x$ because of the hidden constraint.

We have an upper bound, but now we want the best upper bound. We want

$$t \leq \min_\lambda \max_x x^T (Q - Diag(\lambda))x + \lambda^T e$$

where $e = (1, 1, \ldots, 1)^T$. We know that there's a hidden constraint where $Q - Diag(\lambda)$ is NSD, so we might as well add it as part of the minimization criterion. The max is now 0 at $x = 0$. So we can just throw it away, so, this is equal to

$$\min \lambda^T e$$

subject to $Diag(\lambda) \geq Q$ (this is called the Löner partial order). Out pops an SDP. Interior point methods can be extended from LP to SDP.

But recall, we haven't solved the original problem. Max-cut problem is an NP-hard problem. Goemans and Williamson proved that if you take this problem and do a randomization process, the worst you can do is an approx 14% error, which is amazing for a combinatorial optimization relaxation problem. In practice, you get 97% of the optimal solution.

The idea is you take the discrete set and lift it into matrix space. Each point on the n-cube, for example, is lifted into matrix space. This is one example of max-cut where SDP works well. The difficulty is the lifting, where you potentially square the number of variables.

As you go from LP to SDP, the number of iterations is about the same. For LP, you can take on billions of variables. For SDP, you can go to 80000 max-cut nodes.

General Nonlinear Programming: The problem is to minimize $f(x)$ subject to $g_k(x) \leq b_k$. Can allow for equality constraints too. Mathematicians like to use $\leq$ and people in computer science community like the $\geq$ due to the LP duality. Another form is bound constraints on the variables: Minimize subject to $c(x) = 0, l \leq x \leq u$.

The main thing to point out is the Lagrangian again:

$$L(x, y) = f(x) + \sum_i y_i c_i(x)$$

Here, the Lagrange multipliers are y_i instead of λ_i. The purpose of the Lagrange multipliers is that the original problem is too hard to solve. By putting these multipliers together with the objective, again we have a relaxation. Any minimum of the original problem is a stationary point of the second problem. The condition is not sufficient: but we search for stationary points.

There are second-order conditions which provide sufficient conditions. Some of the approaches for solving these include: sequential quadratic programming (uses the local quadratic approximation), reduced-gradient methods, and some interior point methods.

There are algorithms called sequential linear approximation. They use a linearization of the constraints when the constraints are too hard to handle. We must develop a model that we can handle. Solve an LP, and it will hopefully give you the right search direction.

The rough idea of optimization: The functions are too complicated to handle as is, so you make an approximation, usually by Taylor series. There are subtleties to know that you are improving the objective and the constraints.

Infinite Dimensional (Control) Programming: This is an optimal contral problem. Imagine you want the best trajectory (minimum energy) for a rocket ship to get to the moon.

$$\mu_0 = min, J(u) = \frac{1}{2}||u(t)||^2, s.t.x'(t) = A(t)x(t)+b(t)u(t), x(t_0) = x_0, x(t_1) \geq c.$$

Can use the fundamental solution matrix Φ. Every solution $x(t_1)$ can be written as the sum of $\Phi(t_1, t_0)x(t_0)$ and an integral operator $\mathcal{K}_u$.

We now have a **convex program**

$$\min J(k) s.t. \mathcal{K}u \geq d$$

Here, d is finite-dimensional. We're going to again to a Lagrangian relaxation. We add the constraint using λ of same dimension as d.

$$\mu_0 = \max_{\lambda \geq 0} \min_u \{J(u) + \lambda^T(d - \mathcal{K}u)\}$$

As before, this finds a lower bound, so we look for a best lower bound. We have a hidden constraint here as well. And $J(u)$ is a quadratic, so we can explicitly solve for u. The solution for u is

$$u_*(t) = \lambda_*^T \Phi(t_1, t) b(t).$$

$u_*(t)$ is a critical point. Because our function is convex, we're guaranteed that our solution is a minimum.

The $\mathcal{K}$ can come to the other side as an adjoint. We end up with a simple finite-dimensional QP:

$$\mu_0 = \max_{\lambda \geq 0} \lambda^T Q \lambda + \lambda^T d$$

Here, by using Lagrangian duality, we don't need to discretize here. This is what is done for a lot of real-life problems.

In this problem, we have a zero duality gap. When you don't have a zero duality gap, you won't solve your original problem. This example is from Luenberger.

3 MatLAB

Check out the video on the origins of MatLAB, with Cleve Moler.

We are going to use MOSEK, so you have to add it to your path. Find `startup.m` and you can add the lines:

```
addpath /usr/msri/mathsw/mosek/4/toolbox/examp -end
addpath /usr/msri/mathsw/mosek/4/toolbox/r2006b -end
```

Create a simple vector

```
a = [1 2 3 4 5 6 4 3 4 5]
```

Can add 2 to every element of a:

```
b = a + 2
```

Plotting is easy. `plot b` made the line and `grid on` made the grid. Next is a bar graph. There are interactive (as opposed to command-line) ways to get many of these features to work too. Can create matrices:

```
A = [1 2 0; 2 5 -1; 4 10 -1]
```

turns into

$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 5 & -1 \\ 4 & 10 & -1 \end{pmatrix}$$

Can take transpose, or multiply matrices together. To do point-wise multiplication, write

```
C = A .* B
```

This is called the *Hadamard* product. One can use a Hadamard product to represent weights.

Finding inverses is easy:

```
X = inv(A)
```

Can find eigenvalues with `eig(A)`. Can do an SVD decomposition of a matrix A by running `svd(A)`. (Aside on SVD.)

`poly(A)` will give the coefficients of the characteristic polynomial $p_A(t)$ of a square matrix A.

`conv` produces a convolution.

Within the GUI, you can right-click on a vector to do things (such as plot) with it. Without the GUI workspace, you can type `whos`. For he commands loaded, you can type `what`. You can type `clear` to clear your workspace. You can work with complex arithmetic. Type `clear A` to just get rid of `A`.

One of the nice things about MatLAB is solving a system of equations. Solve by typing `x = A\b`. This works by Gaussian elimination. We have a demo here and our first assignment. We examine `Asolve.m`. We will solve using `inv`, then with the slash, then solve by LU, and also with linsolve. Running, we had

```
>> Asolve


Generate random data for A,x and b=Ax, with n=  2000

Solve by x = inv(A)*b:
Elapsed time is 7.466549 seconds.
residual 3.402e-09
error 8.0865e-11

Solve by x = A\b:
Elapsed time is 3.228165 seconds.
residual 1.6859e-10
error 2.2651e-11

Solve by [L,U,P]=lu(A); x=U\(L\b):
Elapsed time is 2.913590 seconds.
residual 1.6859e-10
error 2.2651e-11

linsolve with [L,U,P]=lu(A):
Elapsed time is 2.891677 seconds.
residual 1.6859e-10
error 2.2651e-11
```

We can use `runAS.m`, which does the same thing but starts up a profile. See this file:

```
clear all
profile on
Asolve
profile report
```

This program helps to show the bottlenecks in your computation. Then we have *Aoneseps.m*. This script will make a special matrix.

$$A = \begin{pmatrix} 1 & & & & 1 \\ & 1 & & 0 & 1 \\ & & 1 & & 1 \\ & -1 & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}$$

Then, we'll shake it with a small ϵ:

```
Aeps = A + pert
```

Then, we'll look at `x = A\b` to see $||Ax-b||$ and `xeps = Aeps\b` and examine $||Axeps - b||$.

Type

```
A = Aoneseps(30);
```

Why the discrepancy? The condition number of A was 40. Condition number 40 says that you may lose one decimal of accuracy, so this problem is well-conditioned.

A small change to A doesn't change the eigenvalues of A by much, so it's not about the condition number.

Gaussian elimination is not a stable process. Even for a 2 × 2 matrix

$$\begin{pmatrix} \epsilon & * \\ * & L \end{pmatrix}$$

Instead of pivoting on the small number, pivot on the large number.

MatLAB allows you to use Maple in MatLAB. You can use MatLAB syntax to use Maple. Within the GUI help, see Toolboxes, then Symbolic Math.

Let's consider the 5 × 5 (ill-conditioned) Hilbert matrix.

```
H = sym(hilb(5))
```

H is exact, so there are no decimal places. You get exact determinants. You can get the characteristic polynomial, the inverse, factor.

Can show off using a symbol t, and creating a matrix that is a function of t. Other than `pretty` you can get MatLAB to give you the LaTeX of something by typing `latex` of something.

Can type `keyboard` within your MatLAB script to get a new prompt `K>>` where you can run commands from your code break.

Partial pivoting is not stable. Total pivoting is stable, but is too slow. Total pivoting won't break this problem that we have, so that's a big hint for the exercise. MatLAB has PCG, an iterative algorithm.

This perturbation analysis has become an entire area of research.

Discussion of Assignment 1: In the Gaussian elimination, the last column will have the (sorted) powers of 2. The problem in partial pivoting is that the absolute value of all of these numbers is 1. So, it doesn't do any partial pivoting. With the perturbation, you will be changing rows, and partial pivoting does something, so you no longer go through the row in order.

Iterative methods work in a system $Ax = b$, with A sparse. Iterative methods are based on taking a matrix-vector operation. In fact, if A is PSD, then the iterative method (the Conjugate Gradient method) is based on:

$$\min_x \frac{1}{2}||Ax - b||^2$$

The iterative methods don't try to solve the problem exactly, they just try to make the residual small. The process of preconditioning tries to get the eigenvalues close together.

[There will be more projects, on various topics: regularity and inverses, dimension reduction, statistics, Euclidean distance problem and protein folding.]

4 MatLAB and MOSEK

MOSEK is free for students, and competes with CPLEX. A guided tour is available as a link. MOSEK does not do SDP yet.

[The startup file is now called `mystartup.m`.]

If you add MOSEK, it will take control of several commands, so the optim tool won't work due to clash of keywords.

[SeDuMi is free from McMaster University. "Let SeDuMi seduce you."]

4.1 Linear Programming

Let's do a brief introduction to the simplex method and linear programming. One famous problem is the "diet problem": Suppose we have n foods and m nutrients. For example, suppose that the foods are carrots, liver, etc. Suppose the nutrients are Vitamin A, Vitamin B, calories, etc.

Let A_{ij} be the number of units of nutrient i per unit of food j. The other data that we have is b_i, which is the minimum number of units of nutrient i in our diet. All that's left is c_j, which is the cost per unit of food j. Our decision variable: Find x_j, the number of units of food j in the optimal diet.

The primal problem (P) is to

$$p^* = \text{Minimize } c^T x = \sum_j c_j x_j \text{ subject to } Ax \geq b, x \geq 0.$$

Of course we want $x \geq 0$, we're not bulimic. So, there's nothing here for taste!

George Dantzig visited John Von Neumann at Princeton. Dantzig got to visit and discussed this problem. Von Neumann developed game theory, C^*-algebras, and something else. So, he explained game theory, and duality. This led Dantzig to the simplex method.

In game theory, we have a player X. Maybe he's a dietician, and there's a pay off function. And there's a player Y. The payoff function is a Lagrangian:

$$L(x, y) = c^T x + y^T (b - Ax)$$

The dietician is going to pay for the diet $(c^T x)$, but he's also going to pay for $y^T(b - Ax)$. $b - Ax$ is excess nutrients. So y has to be dollars per nutrient. (Who is player Y?) So,

$$p^* = \min_{x \geq 0} \max_{y \geq 0} L(x, y)$$

Why? There's a hidden constraint: $b - Ax \leq 0$. In other words,

$$\max_{y \geq 0} L(x, y) = \begin{cases} c^T x & \text{, if } Ax \geq b \\ +\infty & \text{, otherwise} \end{cases}$$

If you're infeasible, it's like having an automatic penalty, and so you want to avoid that. The non-negative orthant is a self-dual cone. The y values are the Lagrange multipliers. The y tell us the value of the nutrients.

What's the dual problem? If we put the min before the max, we get this obvious relationship (called Weak Duality):

$$p^* \geq d^* = \max_{y \geq 0} \min_{x \geq 0} L(x, y) = b^T y + x^T (c - A^T y)$$

Really, what we're doing with transpose is an adjoint. Recall $\langle y, Ax \rangle = \langle A^* y, x \rangle$.

Now, what's the hidden constraint? Because we don't want $\min L(x, y)$ to go to negative infinity, no component of $c - A^T y$ can be negative. Thus, $c - A^T y \geq 0$. Thus,

$$d^* = \max\{b^T y : A^T f \leq c, y \geq 0\}$$

This is the dual problem, and this is what led Dantzig to the simplex method. [Player Y is a pill salesman. Let's take a look at the units of y. Remember, it's a cost per nutrient. So, he has to decide how much the pills are worth. He's trying to give you a carrot substitute.]

For linear programming, $p^* = d^*$, if feasible, and both are attained. This is called *strong duality.*

The purpose of this exercise was to show you the hidden constraint, which often appears from Lagrangian relaxation.

$$\begin{array}{rrl} \text{Minimize} & & x_1 + 2x_2 \\ \text{Subject to} & 4 & \leq x_1 + x_3 \leq 6 \\ & 1 & \leq x_1 + x_2 \\ & 0 & \leq x_1, x_2, x_3 \end{array}$$

We can put this in a standard form:

$$\begin{array}{rl} \text{Minimize} & x_1 + 2x_2 \\ \text{Subject to} & x_1 + x_3 \leq 6 \\ & -x_1 - x_3 \leq -4 \\ & -x_1 - x_2 \leq -1 \\ & x \geq 0 \end{array}$$

Now we have $Ax \leq b$. Each one of the constraints in $Ax \leq b$ will have its own Lagrange multiplier. The dual will look like

$$\max\{b^T y : A^t = c, y \geq 0\}$$

This one here is called *standard form.*

The term *duality gap* is used in several ways: either for your specific x, y or for x^*, y^*. The simplex method works because of duality.

The MOSEK function `msklpopt` has its own form that has upper and lower bounds for Ax and for x. Input c cost and a the A matrix. Each line in the iteration output is a step of Newton's method. What we're trying to do is to get the primal and dual objective values to be equal. The number of iterations grows very slowly. This is an interior point method. The solution is "purified" at the end by finding the correct primal basis.

We argue that an optimum is found at a vertex. There may be a tie, but an optimum is found at a corner point. That solution is called a basic feasible solution. Then the program outputs the answer. `sol` is a new variable in our MatLAB workspace. Find the answer in `sol.bas.xx`.

There are two dual variables; `sol.bas.y`. The dual problem is solved at the same time.

How do we go from an infeasible point to a feasible point? There are some equations that need to be satisfied by feasible points. As we iterate through the equations, we force some of the equations to be satisfied.

We can try to prove complementary slackness. We can add a slack variable z:

$$Ax + z = b, z \geq 0.$$

Proposition 4.1. *Note that x, y is optimal if and only if $Ax + z = b, z \geq 0$ (primal feasibility), $A^T y = c, y \geq 0$ (dual feasibility), and $c^T - b^T y = 0$ (no gap).*

Proof. Follow the string of equalities: $0 = c^T x - b^T y = c^T x - (ax + z)^T y = x^T(c - A^T y) - z^T y$ if and only if $z^T y = 0$ (called complementary slackness) which is true iff $z \circ y = 0$ (Hadamard product). □

We have $2m+n$ variables, and we can replace the "zero gap" constraint with complementary slackness: $z \circ y = 0$. There are $tm + n$ equations. We can call this entire system (make it homegenous in zero) a function $F(x, y, z) = 0$ (but not the inequalities). Note,

$$F : \mathbb{R}^{2m+n} \to \mathbb{R}^{2m+n}$$

We solve it by Newton's method.

Can also type `help sedumi`. The help tells you about the standard form that you need to put the problem in.

We can also solve using `mosekopt`. Let's look at the example `lo2.m`.

[There are many different kinds of duals.]

In this problem, we note that $x_1 + 2x_2 = x_2 + (x_1 + x_2) \geq x_2 + 1$, and this gave us (in this example) a simple proof for this problem.

4.2 Convex Quadratic Optimization

In these packages, they always tell you that the Q has to be PSD. They only have to be PSD on the feasible set.

Proposition 4.2. *A matrix Q is positive definite if and only if all leading principal minors (consider all upper-left square submatrices) are positive.*

This does not extend to positive semidefinite. For PSD, you have to check **all** minors. We can run the example `qo1.m`.

The Gershgorin circle theorem identifies a region in the complex plane that contains all the eigenvalues of a complex square matrix.

Proposition 4.3. *(Gerşgorin) The eigenvalues lie within the discs centered at the diagonal entries a_{ii}, with radius r_i, where*

$$r_i = \sum_{i \neq j}^{n} |a_{ij}|$$

This can be proved from diagonal dominance.

Quadratic programs have a Lagrangian dual, and it can be found the same way as for linear programs.

The quadratic program

$$\begin{array}{rc} Minimize & \frac{1}{2}x^T Q x + c^T x \\ Subj & Ax = b \\ & x \geq 0 \end{array}$$

We have

$$p^* = \min_{x \geq 0} \max_y \frac{1}{2} x^T Q x + c^T x + y^T (b - Ax) = L(x, y)$$

And this is greater than or equal to the dual:

$$\geq d^* = \max_y \min_{x \geq 0} L(x, y)$$

We have to do things a little bit differently.

$$\max_{y; w \geq 0} \min_{x \geq 0} L(x, y) - w^T x$$

Now that we do this, we can use the hidden constraint idea. We have a convex quadratic. There will be a minimum iff the gradient is zero. This is called the *Wolfe*[1] *dual*. We add the (hidden) constraint underneath the max:

$$0 = \nabla_x L(x, y) - w$$

If we have a quadratic $q(x)$ (convex), and we have the Hessian $Q \succeq 0$, then we have

$$\mu^* = \min q(x)$$

Then, $\mu^* > -\infty$ if and only if μ^* is attained (ie, $\exists\ \overline{x}$ such that $\mu^* = q(\overline{x})$), and this happens iff $0 = \nabla q(\overline{x})$.

If our objective is $q(x)$, what can go wrong? The gradient is just $0 = \nabla q(x) = Qx + c$. If $Q \succ 0$, then there is always an x given by

$$x = -Q^{-1} c$$

But if $Q \succeq 0$ is singular, we have to be lucky, and we need c to be in the range of Q.

We also have `qo2.m`, which shows how to build things up with a MatLAB structure. Then, we can solve using `mosekopt`.

Duality is hiding behind all of our optimization problems. Go back to the idea of the dietician and the pill-maker. There are certain algorithms that don't work like this: they stay primal feasible and use some kind of stochastic process.

Whenever you're dealing with a constraint, you can always penalize them in this way using a Lagrange multiplier. That simple idea leads to a duality theory. If you have a non-constrained problem, you don't talk about this primal-dual stuff.

We can look at an example using `mosekopt` as in the guided tour, section 7.5.3.

We go on to convex problems, and talk about the same idea. The interior point methods are being used for general non-linear problems: it's being used for all problems, all models.

[1] Phillip Wolfe was a mathematician at IBM. He stopped doing mathematics, started cleaning offices, and finally got fired.

`sprandsym` will generate a random sparse symmetric matrix in MatLAB. Use `spy(A)` to look at a schematic picture of the matrix.

```
r(1:n+1:end) = zeros(100,1); % replace diagonals of r with 0s.
nnz(r) % shows number of nonzeroes.
spy(r)
```

MatLAB's original command for linear programming is `linprog`. MOSEK takes this over, but the syntax is the same. There's also `quadprog`: See the syntax in the MatLAB help again. Type `help optimset` to change options for this problem. You get a structure of options in your workspace.

4.3 Conic Optimization

We saw one example where the relaxation for Max-Cut gave us an SDP. SDP are a special case of conic optimization problems. We have a set $C \subset \mathbb{R}^n$. Then C is a *cone* if $\lambda C \subset C$ for all $\lambda > 0$. Sometimes, you can include the point zero, as most books do. This means that C contains all half rays.

A cone C is a *convex cone* if $C + C \subset C$. Equivalently, C is a convex set. In three dimensions, you can have polyhedral cones or non-polyhedral cones. A cone is *polyhedral* if can be written as

$$C = \{x : Ax \leq 0\}$$

This is the intersection of [linear] halfspaces. An example of a polyhedral cone that we have been working with is the non-negative orthant $\mathbb{R}^n_+$. This orthant gives us a partial order. We say $x \geq 0$ element-wise iff $x \in \mathbb{R}^n_+$. Then, we can say that $x \geq y$ means $x - y \geq 0$. Often, the relation has the subscript: $\geq_{\mathbb{R}^n_+}$.

Now, if we take K to be any closed convex cone (c.c.c. for short), then we say $x \geq_K 0$ to mean $x \in K$, and $x \geq_K y$ means $x - y \geq_K 0$ or $x - y \in K$.

A cone K is *pointed* if $K \cap -K = \{0\}$. This seems to be a requirement for forming a partial order. Let's just assume that our cone is pointed for now.

We can take a general optimization problem

$$\begin{array}{rll} min & f(x) & \\ s.t. & g_k(x) \leq 0 & k = 1, \ldots, m \\ & h_j(x) = 0 & j = 1, \ldots p \end{array}$$

we can write $g(x) \leq_{\mathbb{R}^m_+}$ and $h(x) \leq_{\{0\}^p} 0$. And these can be combined:

$$k(x) = \begin{pmatrix} g(x) \\ h(x) \end{pmatrix}$$

where $K = \mathbb{R}^m_+ \otimes \{0\}^p$, $k(x) \leq_K 0$.

Now, we can play this game again, with player X. We have min $f(x)$ s.t. $g(x) \leq_K 0$, a K-convex function with $x \in \Omega$, the convex set. Again, we have

$$\min_{x \in \Omega} \max_y L(x, y) := f(x) + \langle y, g(x) \rangle$$

The feasible set is the set $\{x \in \Omega : g(x) \leq_K 0\}$. That is, $g(x) \in -K$.

We could have: minimize the trace of $AXBX^T + CX^T$ and the constraints are: $g(X) = X^T X - I \leq_{S_+^n} 0$, where S_+^n is the cone of PSD matrices. Thus, this is a function

$$g : \mathbb{R}^{n \times n} \to S^n$$

The Lagrange multipliers[2] are sitting in the space of symmetric matrices, so the Lagrange multipliers are **matrices**!

Within the max above, where does y live? In the dual cone (or polar cone). I like the notation K^+ as opposed to K^*. This is:

$$K^+ := \{\varphi \in X^* : \langle \varphi, k \rangle \geq 0 \ \ \forall \ k \in K\}$$

This is all vectors making angles less than 90 degrees to all vectors in the cone K. In $\mathbb{R}^2$ is easiest to see. Note, we have

$$\min_{x \in \Omega} \max_{y \in K^+} f(x) + \langle y, g(x) \rangle$$

for the min-max statement.

Often in optimization problems, the collection of constraints are split up into two kinds: the hard and easy ones. We should think of everything as infs and sups, even in finite dimensions.

In the guided tour, we consider three kinds of cones:

- $\mathbb{R}_+^n$ cones
- Quadratic cone (Lorentz "ice cream" cone)
- Rotated quadratic cone

All of these cones are self-dual:

$$K = K^+.$$

They also have non-empty interior, and they are proper, closed, and convex.

Subsection 7.6.2 gives us an example problem.

$$\begin{aligned} min \quad & x_5 + x_6 \\ s.t. \quad & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0, \\ & x_5 \geq \sqrt{x_1^2 + x_3^2} \\ & x_6 \geq \sqrt{x_3^2 + x_4^2} \end{aligned}$$

A function g is K-convex if

$$g\left[\alpha x + (1-\alpha) y\right] \leq_K \alpha g(x) + (1-\alpha) g(y)$$

[2] Lagrange multipliers are due to his supervisor Euler. Did you know that l'Hôpital bought his rule?

for all $0 \le \alpha \le 1$ and for all x and y. Let's see how this function relates to ordinary convexity. This is the same as saying that

$$g\left[\alpha x + (1-\alpha)y\right] - \alpha g(x) - (1-\alpha)g(y) \in -K = -K^{++}$$

This is a generalization of Jensen's inequality.

Because we are in K^{++}, we can say: Let $\varphi \in K^{+}$. Then define

$$g_{\varphi} : X \to \mathbb{R}$$

by the rule

$$g_{\varphi}(x) = \langle \varphi, g(\alpha x + (1-\alpha)y) - \alpha g(x) - (1-\alpha)g(y)\rangle \le 0$$

for all $0 \le \alpha \le 1$.

This is saying that the function $g_{\varphi}(x) := \langle \phi, g(x)\rangle$ is convex for all $\varphi \in K^{+}$. So, we can phrase K-convexity in terms of ordinary convexity. So instead of writing $g(x) \le_K 0$ with the constraint: $g_{\varphi}(x) \le 0$ for all $\varphi \in K^{+}$.

Let's follow the example `cqo1.m`. A linear objective function will find a solution on the boundary, so we saw that this solution has optimum on the boundary of the cone. The only other possible place was on the non-negative orthant.

4.4 Quadratic and Conic Optimization

MOSEK can handle quadratic constraints. The constraints are conic. Most of these, you can replace these by a conic. `cqo2.m` is an example problem.

4.5 Dual Cones

Now, we can see the role of the dual cone C^*. Examples of the dual cones are on the self-guided tour.

4.6 Setting accuracy parameters

You can set these options using MatLAB commands.

4.7 Linear least squares and related norm

Can use the QP solver to solve the linear least squares problem on the $||\cdot||_2$-norm. In the examples, we look at the ∞-norm.

In the 1-norm, you'll get lots of zeroes. Often times you want that. In portfolios, you want the exact opposite of this behavior. They're trying to avoid the effects of the 1-norm by adding other constraints. But, they can't avoid this because they're using CPLEX.

4.8 Mixed Integer Problems

We can take a look mixed-integer constraints. If you build airplanes, you can't build have an airplane. Integer problems are very important, but very hard to solve. Typically involve branch-and-bound methods, and cutting planes. Typically they start with an LP relaxation.

We have an example program `milo1.m`. These are very hard. You can find problems with 15 variables that you can't solve.

One of the interesting problems in this area is the *quadratic assignment problem.* Suppose you had to build a university, but you don't know where you should build the buildings. There's a cost in terms of building separation. It's quadratic because it's number of people times a distance!

The NEOS site has problems up to size 9. Up until 1980, no one could solve anything beyond 15 facilities and 15 locations. In 1980, someone did $n = 16$, and three or four years ago, $n = 30$ was solved, related to VLSI design. This problem is also related to chip design.

One $n = 30$ was solved on the system CONDOR in Minnesota. CONDOR steals cycles around the world. They farmed out these jobs to thousands and thousands of computers around the world. It was a combination of theory, and it was SDP relaxation that provided a stronger bound. The bound was the big problem in these branch-and-bound processes.

All 37 of us should send our QAP problem to NEOS. Maybe I'll get an e-mail from them complaining.

4.9 Large Scale QP

In `spqp.m`, Let's take blocks, and generate a random problem. We'll solve this using `quadprog`.

5 A word on convexity

A function h is convex if for all $0 \leq \alpha \leq 1$,

$$h(\alpha x + (1-\alpha)y \leq \alpha h(x) + (1-\alpha)h(y)$$

The nice thing about convexity is that we can just deal with a single parameter t. If we draw the secant line, then we have that the function lives below the secant line. In calculus, we call these *concave up.*

Another thing about convex functions: If we look at a level curve. The set

$$L_\beta = \{x : h(x) \leq \beta\}$$

is a convex set.

For convex functions, the directions of decrease have derivative less than zero.

Note, there are functions called *pseudoconvex*, and function that are *quasiconvex.* These don't have as nice of a property in terms of the calculus. Their preimages are still convex. Puesdoconvex functions come up a lot in economics (e.g., the sharp ratio). Quasiconvex functions also come up in economics.

Then, if we go to the generalization with K-convex:

$$G(x) \leq_K 0$$

Then, these have the same properties. It's just that you have to use the appropriate inner product. We also are careful with what kinds of objects our Lagrange multipliers are.

If $g(x)$ is a vector, we just have $y^T g(x)$. So, we have $y^T g \geq 0$ for all $y \geq 0$. The set of convex functions forms a convex cone.

In the setting with G, the set

$$\{x | G(x) \leq_K 0\} = \{x | \langle \varphi, G(x) \rangle \leq 0 \;\; \forall \; \varphi \in K^+\} = \bigcap_{\varphi \in K^+} \{x | \langle \varphi, G(x) \rangle \geq 0\}$$

Suppose you have a quadratic $q(x) = \frac{1}{2} x^T Q x + b^T x$. Then q is convex iff $Q \succeq 0$. Then, we take a look at $X^T X - I \leq_{S^n_+} 0$. It is not obvious at all that this is K-convex. We'll leave this as an exercise to assign at the end of this week.

Another project I'm going to add: If you remember, I wrote down an infinite-dimensional example. Due to convexity, the infinite-dimensional problem was changed into a very nice finite-dimensional problem. So, I'll record the details of this, but those of you who may be interested should talk to me afterwards. The question is that we have some interpolation conditions and we want to find a function $x(t)$ on some interval, say, $[0, 1]$. x should satisfy some interpolation conditions, e.g.:

$$\int_0^1 x(t) \psi_i(t) dt = b_i, \quad i = 1, \ldots, n$$

for given $\psi_i(t)$. We also want that $x(t)$ is convex. This is called *best convex interpolant*. We want to find a function, but we only have a finite number of constraints.

I'll provide some details, but in the end, you're going to end up with the best cubic spline, the convex interpolant for this problem. It's very easy to work with the dual for this problem, and easy to get a solution from Newton's method, of course. Note that $x \in L_2[0, 1]$ must hold. In the end, by doing a best-interpolation, we're going to be minimizing the norm $||x(t)||$, subject to some appropriate details.

5.1 Large scale problems

We're going to modify `spqp.m`. From the GUI, we can remove MOSEK from the path. To check if MOSEK is in the path, we can try to run `mosekopt`. Now, `quadprog` is taken over by MatLAB. Even a small problem (which took no time on MOSEK) took 1.644 seconds. At $k = 10, t = 40$, the program embedded in MatLAB had no trouble. Let's try larger k and t.

MatLAB runs this on an “active set” method. This is akin to the simplex method traversing from vertex to vertex. MatLAB has `optimtool` which starts up a GUI interface to the optimization tools.

MatLAB does *automatic differentiation*. Maple does *symbolic differentiation*. Automatic differentiation: If you have a black box (a file with computer code), the function differentiates your software, and it does it efficiently.

You can download the automatic differentiation tools. They're available online as public domain. People made errors entering the differentiations.

6 Convex sets

Convex analysis is a beautiful area, studied independently. This is in the core of our interior point revolution.

6.1 Introduction

The simplest kind of set is an affine set. An affine set contains the line through any two distinct points in the set. This affine combination is an unbounded version of a convex combination. Our familiar example is the solution set of linear equations. In fact, every affine set can be described this way.

A convex set contains all line segments. For the points $x_1, \dots, x_k$, a convex combination is the set of all points

$$x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k$$

where $\sum \theta_i = 1$, $\theta_i \geq 0$. The *convex hull* of the set S is the smallest convex set containing S. A convex hull does not have to be a closed set. Consider a polygon, for example, missing an edge.

We can look at convex cones. Here, we allow the scalars to be any nonnegative, and we don't worry about summing to 1.

A hyperplane is a set of the form

$$\{x | a^T x = b\} (a \neq 0)$$

A halfspace is a set of the form

$$\{x | a^T x \leq b\} (a \neq 0)$$

A hyperplane is the boundary of a halfspace. Halfspaces come with a normal vector (that we will label a). Hyperplanes are affine and convex. Halfspaces are convex.

The *(Euclidean) ball* with center x_c and radius r can be written in two ways:

$$B(x_c, r) = \{x \mid ||x - x_c||_2 \leq r\} = \{x_c + ru \mid ||u||_2 \leq 1\}$$

An *ellipsoid* is a set of the form

$$\{x \mid (x - x_c)^T P^{-1} (x - x_c) \leq 1\}$$

with $P \in \mathbf{S}^n_{++}$ (ie, P is symmetric positive definite).

If $W^T W \succeq 0$ if and only if W is non-singular. We're really looking at $||u||^2_W = ||Wu||^2 = u^T(W^T W)u$. Here, we set $P^{-1} = W^T W$. The other representation is

$$\{x_c + Au \mid ||u||_2 \leq 1\}$$

where A is square and non-singular. (The A is the scale.)

A norm ball with center x_c and radius r is $\{x : ||x - x_c|| \leq r\}$. A norm cone is $\{(x,t) : ||x|| \leq t\}$. This is the second-order cone when the Euclidean norm is used. MOSEK can handle these cones.

6.2 Polyhedra

A polyhedral set is the intersection of a finite number of halfspaces.

6.3 Positive semidefinite cone

- $\mathbf{S}^n$ is the set of symmetric $n \times n$ matrices.
- $\mathbf{S}^n_+$ is the positive semidefinite matrices. This is a convex cone.
- $\mathbf{S}^n_{++}$ is the positive definite matrices.

For 2 × 2 matrices, this looks like an ice cream cone. This can be seen by the definition for positive semidefinite matrices using the inner product with I.

The symmetric 2 × 2 matrix

$$\begin{pmatrix} x & z \\ z & y \end{pmatrix}$$

is positive semidefinite under what conditions? $\langle X, I\rangle = tr(XI) = tr(X) = x + y$. Thus,

$$\cos\theta_{X,I} = \frac{\langle X, I\rangle}{||X||\ ||I||}.$$

$\Rightarrow X \succeq 0$ iff $xy - z^2 \geq 0$. We have here all matrices that make an angle of less that 45 degrees with I. This is why this cone is self-dual/polar.

Why can't we use this argument in higher dimensions? We still have the angle conditions, don't we? It doesn't characterize positive semidefiniteness in higher dimension.

What can we say if $\theta_{X,I} \leq 45$ degrees? That is $tr(X) \geq 0$ and the cosine angle formula gives $\frac{\langle X,I\rangle}{||X||||I||} \geq \frac{1}{\sqrt{2}}$. Squaring both sides, we get

$$(tr(X)^2) \geq \frac{(tr(X^2))tr(I)}{2}$$

What does it mean that the trace of X is greater than or equal to zero?

$$(tr(X))^2 \geq ||X||^2||I||^2 = (trX^2)n$$

This is the same as saying that

$$\frac{(tr(X)^2)}{n^2} \geq \frac{tr(X^2)}{n}$$

Can we prove that the semidefinite cone is self-polar? $\mathbf{S}^n_+$ is the cone of positive semidefinite matrices. Now,

$$(\mathbf{S}^{\mathbf{n}}_{+})^{+} = \{Y \in S^n : \langle X, Y \rangle = tr(X^TY) = tr(XY) \geq 0 \ \forall \ X \succeq 0\}$$

Let's show that this is equal to S^n_+. Let's start with a matrix $Y \succeq 0$ and $X \succeq 0$. Then, $tr(XY) =?$. You can always do a Cholesky decomposition $X = LL^T$.

$Y \geq 0$ iff $x^TYx \geq 0 \forall x$. Thus, $y^TL^TFLy) = (Ly)^TY(Ly) \geq 0$. So, $S^n_+ \subset (S^n_+)^+$.

To show $(S^n_+)^+ \subset S^n_+$, let $Y \in (S^n_+)^+$, that is, $tr(YX) \geq 0$ for all $X \geq 0$. By contradiction, suppose $Y = PD_YP^T$ and

$$D_Y = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

And suppose that the smallest eigenvalue $\lambda_n < 0$.

Let

$$X = P \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} P^T \geq 0$$

Then $tr(YX) = tr(PD_YP^TP)[0;1]P^T = tr(D_Y)[0;1] = \lambda_n < 0$, which is a contradiction.

Is this still self-dual if you change the inner product?

In the boundary of the 2×2 cone, each ray corresponds to (the multiples of) a matrix of rank 1. we can see this by the equality we set. Indeed, if our matrix is

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

then we have the equation (as opposed to inequality) $ac = b^2$ and $a \geq 0$, and $c \geq 0$. Then we have the set

$$\left\{ \alpha \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} : \alpha \geq 0 \right\}$$

which is a one-dimensional cone. So, we can write this set as

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \alpha \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}^T$$

for $\alpha \in S_+$, and we can do this in general. So F is a "face" of S^n_+ if and only if $F = R(S^r_+)R^T$ where r is equal to the rank of any matrix $X \in relint(F)$ and

R are the eigenvectors (orthogonal) corresponding to non-zero eigenvalues. (If a problem has zero eigenvalues, we can go down into a lower dimension.) So, every face of the non-negative orthant is the cone of a smaller dimensional cone. For many problems (graph partitioning, etc.) you have to use this property.

6.4 Preserving convexity

What are the operations that preserve convexity. The intersection of two convex sets is convex. Affine functions: taking the affine image of a convex set preserves convexity.

Also, more surprisingly, perspective functions and linear fractional functions preserve convexity.

The image of a convex set S under $f = Ax + b$ is convex. Inverse images are also convex. Examples include scaling, translation, and projection. The solution set of linear matrix inequalities are convex sets. They hyperbolic cone $\{x \mid x^T P x \leq (c^T x)^2, c^T x \geq 0\}$ (with $P \in S_+^n$).

Then, there are perspective functions:

$$P : \mathbb{R}^{n+1} \to \mathbb{R}^n$$

by

$$P(x,t) = \frac{x}{t}$$

Images and inverse images of convex sets under perspective are convex. The same is true for linear-fractional functions $f : \mathbb{R}^n \to \mathbb{R}^m$ defined as

$$f(x) = \frac{Ax + b}{c^T x + d}$$

The images and inverse images of convex sets are convex. As an aside, we think of the epigraphs of pseudo-convex sets.

6.5 Generalized Inequalities

A convex cone K is *proper* if it is closed, solid (non-empty interior), and pointed. The non-negative polynomials on $[0,1]$

$$K = \{x \in \mathbb{R}^n : x_1 + x_2 t + x_3 t^2 + ... \geq 0\}$$

is an example, as are the nonnegative orthant, the PSD cone, and the second-order cone.

We have generalized inequalities with respect to the cone K as $x \leq_K y \Leftrightarrow y - x \in K$. Similiarly,

$$x <_K y \Longleftrightarrow y - x \in int(K)$$

Some examples include componentwise inequality and matrix inequality ($Y - X$ is PSD). The standard linear-addition property holds.

This is not a linear ordering: there are incomparables in general. $x \in S$ is the minimum element of S if $y \in S \Longrightarrow x \leq_K y$. $x \in S$ is a minimal element of S if $y \in S, y \leq x \Longrightarrow y = x$.

6.6 Separating Hyperplane Theorem

In functional analysis this is called the Mazure theorem, and it's a geometric version of the Hahn-Banach theorem. The observation is that if you have two disjoint convex sets C and D, (and say at least one is compact), then there is a hyperplane $\{x|a^Tx = b\}$ separating the two. That is, $a^Tx \leq b$ for all $x \in C$ and $a^Tx \geq b$ for all $x \in D$. Strict separating requires additional assumsptions (such as C is closed and D is a singleton).

How do we prove this special case? Take a point $d \in D$ and consider the ball centered at c the singleton of radius $d(d, c)$. Consider the intersection of the ball of the circle with D, and call this E. Then

$$\min_{c \in C} ||c - e||$$

then f is convex on $\mathbb{R}^n$. Then, in fact, f is locally Lipschitz. Then Wierstrauss says that there is a solution, that is, a closest point $\overline{d}$ exists. Then the trick is to choose the normal a in the direction of $\overline{cd}$ and let your fixed point be the midpoint of c and d.

Now, you have the proof in the special case, and there's a way of extending this to proving it for two arbitrary convex sets.

There's a famous argumnent called the *supporting hyperplane theorem.* A supporting hyperplane to a set C at the boundary point x_0 is

$$\{x|a^Tx = a^Tx_0\}$$

where $a \neq 0$ and $a^Tx \leq a^Tx_0$ for all $x \in C$. This proves Nathan's claim for yesterday about intersections of hyperplanes.

Proposition 6.1. *(Supporting Hyperplane Theorem) If C is convex, then there exists a supporting hyperplane at every boundary point of C.*

Historically, Farkas was the way duality was proved. This was proved around 1900, and it took three times to prove it. Now, there's essentially a three-line proof. If you don't know how to do something, try using it as a Hail Mary pass.

The polar/dual of a cone is given as

$$K^* = \{y|\ y^Tx \geq 0 \ \ \forall\ x \in K\}$$

As note, the norm-duality comes into play in the definition of our norm cones.

Challenge: prove the lemma.

Proposition 6.2. $K = K^{++}$ *if and only if* K *is a closed convex cone.*

Let's do this as a HW. So, the hint is to use the separating hyperplane theorem.

Proof. Suppose $K = K^{++} = (K^+)^+$. Note that $(K^+)+$ is

$$\bigcap_{\phi \in K^+} \{y \mid \langle y \in \phi \rangle \geq 0\}$$

This is an intersection of closed and bounded sets, so it's a closed and bounded set.

Now suppose K is a closed convex cone. If $k \in K$ and $\phi \in K^+$, then $\langle \phi, k \rangle \geq 0$. That is, $\langle k, \phi \rangle \geq 0 \ \forall \ \phi \in K^+$, thus $k \in (K^+)^+$.

We wish to show $K^{++} \subset K$. Suppose not. Then, there is $y \in K^{++}$ but $y \notin K$. Then, be strict separation, $\exists \ a, \alpha$ such that

$$\langle y, a \rangle < \alpha < \langle k, a \rangle, \quad \forall \ k \in K$$

Since $0 \in K$, we get $\alpha < 0$, so $\langle y, a \rangle < \alpha < 0$. None that

$$\langle k, a \rangle \geq 0 \text{ for all } k \in K \tag{1}$$

since K is a cone. Note that (1) implies $a \in K^+$. But $y \in K^{++}$ and

$$\langle y, a \rangle < \alpha < 0,$$

a contradiction. □

We can use this to give an easy proof of the Farkas lemma.

Lemma 6.3. *(Farkas' Lemma)* $0 = \min\{b^T y : A^T y \geq 0\}$ *if and only if* $A\lambda = b$ *is consistent for some* $\lambda \geq 0$.

To prove this, you do need to know something about the cone that we're talking about here. Let's do this as an assignment also.

What about the second problem: Derive necessary and sufficient conditions for positive semidefinitieness of X in $\mathbf{S}^n$ based on the angle between X and I.

Given $X = X^T$, $X \succeq 0$ iff $\lambda_i(X) \geq 0$ for all i. Let's take

$$Y = P \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} P^T$$

Let's consider the angle θ between X and I.

$$(\cos\theta)^2 = \left(\frac{\langle X, I \rangle}{||X|| \ ||I||} \right)^2 = \frac{(tr(PDP^T))^2}{(tr(PDP^T PDP^T))(n)} = \frac{(\sum \lambda_i)^2}{n(\sum \lambda_i^2)}$$

So, we're taking a look at moments here. We know constant $M_1 = \sum \lambda_i$, $M_2 = \sum \lambda_i^2$. Let's look at:

$$\begin{array}{rl} min & \lambda_1 \\ s.t. & \sum \lambda_i = M_1 \\ & \sum \lambda_i^2 = M_2 \end{array}$$

How do we solve this problem? Apply Lagrange multipliers. This satisfies a constraint qualification. For now, let's just assume this holds. You have to guarantee linear independence of the constraints. So the Lagrangian is (let's use y for the Lagrange multipliers)

$$L(\lambda, y) = \lambda_1 + y_i(\sum(\lambda_i - M_1) + y_2(\sum \lambda_i^2 - M2)$$

If λ^* is optimal, then

$$0 = \nabla_\lambda(L, y) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + y_1 \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + 2y_2 \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_n \end{pmatrix}$$

$y_2 = 0 \Rightarrow y_1 = 0$, a contradiction, so $y_2 \neq 0$. So $\lambda_2 = \lambda_3 = \cdots = \lambda_n$. Substitute into the constraints. We get

$$\begin{aligned} \lambda_1 + (n-1)\lambda &= M_1 \quad , \ \lambda = \lambda_2 = \cdots = \lambda_n \\ \lambda_1^2 + (n-1)\lambda^{=} &M_2 \end{aligned}$$

Let $m = \frac{M_1}{n} = \frac{\sum \lambda_i}{n}$. Then let $s^2 = \frac{M_2}{n} - m^2$, so now we have the mean and the variance. So $\lambda = m + \sqrt{n-1}s$ or $\lambda = m - \frac{1}{\sqrt{n-1}}s$. So we get $m - \frac{1}{\sqrt{n-1}}s \leq \lambda \leq m + \sqrt{n-1}s$

Now, we have a sufficient condition: $m - \frac{s}{\sqrt{n-1}} \geq 0 \implies X \succeq 0$. We can write

$$\begin{aligned} \sqrt{n-1}m &\geq s \\ (n-1)m^2 &\geq s^2 \\ (n-1)\left(\frac{\sum \lambda_i}{n}\right)^2 &\geq \frac{(\sum \lambda_i)^2}{n} - \left(\frac{\sum \lambda_i}{n}\right)^2 \\ n\left(\frac{\sum \lambda_i}{n}\right)^2 &\geq \frac{(\sum \lambda_i)^2}{n} \end{aligned}$$

But this last condition holds

$$\begin{aligned} &\iff (\sum \lambda_i)^2 \geq \sum \lambda_i^2 \\ &\iff (tr(X))^2 \geq (tr(X^2)) \\ &\iff (\cos\theta)^2 = \frac{(tr(X))^2}{||X||^2 n} \geq \frac{1}{n} \end{aligned}$$

So, if we want to guarantee that a matrix is PSD, the angle with I has to get smaller and smaller as n gets bigger.

Is there a big circle that **contains** all PSD matrices? Can we do another optimization problem? We just solved

$$\begin{aligned} max \quad & \theta \\ s.t. \quad & X \succeq 0 \end{aligned}$$

If $X \succeq 0$, then $(\cos\theta)^2 \geq$?

So, we found a lower bound for the smallest eigenvalue of the matrix. The smallest eigenvalue is going to be bounded below by $m - \frac{1}{\sqrt{n-1}}s$.

6.7 Basic Properties of Convex Functions

We already saw the property that any chord between two points in a convex function lies above the graph.

Some examples of convex functions on (subsets of) $\mathbb{R}$:

- affine
- exponential (e^{ax})
- powers x^α
- powers of absolute value: $|x|^p$ for $p \geq 1$
- negative entropy $x \log x$ on $\mathbb{R}_{++}$

Some concave functions:

- affine
- powers
- logarithms

If f is both convex and concave, then f is affine.

On $\mathbb{R}^n$, we have affine functions $f(X) = tr(A^T X) + b$ and norms ($||x||_p$, for $p \geq 1$, including ∞). Examples on convex functions on $m \times n$ real matrices include affine functions and the spectral (maximum singular value) norm function $f(X) = ||X||_2$. The spectral radius $\rho(X)$ is not a valid norm.

We can always use a "restriction to a line" trick for convexity. We can check convexity by just checking a function of one variable. One important example: Let $f : \mathbf{S}^n \to \mathbb{R}$ with $f(X) = \log\det X$, with $\text{dom}(X) = \mathbf{S}^n_{++}$.

We'll have an exercise related to this. Recall that if $X = PDP^T$, then $\sqrt{X} = P\sqrt{D}P^T$. Note that $X = (\sqrt{X})^2$.

Question: Does every matrix have a square root? For a symmetric, you have to use complex numbers. Just go to the 1×1 matrix $[-1]$. Does

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

have a square root?

How would you prove that a convex function has to be continuous, or locally Lipschitz continuous?

Another approach to optimization to give ultimate penalties to certain indicator functions:

$$\delta_c(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}$$

So, we can "replace" the Lagrangian with

$$f(x) + \delta_{\mathbb{R}^m_+}(Ax - b)$$

There is a first-order condition for convexity: it says that a tangent line lies under the graph of the function. This can be proved using a limit of the Jensen inequality. So, the first-order approximation (the Taylor series T_1 approximation) is always a global under-estimator of f.

And, the second-order condition is: We need that the Hessian $\nabla^2 f(x) \succeq 0$. This is if and only if. If you only have $\nabla^2 f(x) \succ 0$ for all $x \in dom(f)$, then f is strictly convex. The converse is not true (consider $f(x) = x^4$ at $x = 0$).

6.8 The "Clown Car Paradox"

Here's a question for you: In $\mathbb{R}^n$, place unit spheres $A_1, \ldots, A_{2^n}$ centered at all points of the form

$$(\pm 1, \pm 1, \ldots, \pm 1).$$

Let C_n be the inscribed sphere (C_n is centered at the origin and tangent to all of the A_i). Let B_n be the (Cartesian) rectangular box $[-2, 2]^n$. Then, as n goes to ∞, what happens to

$$\frac{vol(C_n)}{vol(B_n)}$$

This ratio is going to $+\infty$ as $n \to \infty$. Seemingly paradoxically, when $n \geq 5$ C_n is not contained in the box B_n.

6.9 Patching up earlier work

We have the condition:

$$m - \sqrt{n-1}s \leq \lambda_{min}(X) \leq m - \frac{s}{\sqrt{n-1}} \leq 0$$

Notice that when $n = 2$, the bounds on both sides of $\lambda_{min}(X)$ are the same.

6.10 Back to convex functions...

Some examples are quadratic functions:

$$f(x) = (1/2)x^T P x + q^T x + r$$

and the least-squares objective function $f(x) = ||Ax - b||_2^2$. The constraints here are as well, so LS is a convex program.

Interestingly, a quadratic over a linear $f(x, y)x^2/y$ is convex for $y > 0$.

Another example: log-sum-exp $f(x) = \log\left(\sum_{k=1}^n \exp(x_k)\right)$ is a convex function. Use the Cauchy-Schwarz inequality to prove this.

The geometric mean $f(x) = \left(\prod_{k=1}^n x_k\right)^{1/n}$ on $\mathbb{R}^n_{++}$ is concave. The proof here is similar.

6.11 Graphs of functions and Epigraphs

The *epigraph* of $f : \mathbb{R}^n \to \mathbb{R}$ is a set in one dimension higher, consisting of the graph and all points above.

6.12 Jensen's Inequality

If f is convex, then for $0 \leq \theta \leq 1$,

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

As an extension to probability,

$$f(\mathbb{E}x) \leq \mathbb{E}(f(x))$$

6.13 Proving convexity

Here are typical tools for proving that a function f is convex:

1. Verify the definition (often by restricting to a line)
2. For twice differentiable functions, show $\nabla^2 f(x) \succeq 0$
3. Show that f is obtained from simpler convex functions by operations that preserve convexity:
   - non-negative weighted sum
   - composition with affine function
   - point-wise maximum and minimum: If $f(x,y)$ is convex in x for each $y \in \mathcal{A}$, then
     $$g(x) = \sup_{y \in \mathcal{A}} f(x,y)$$
     is convex.
   - composition
   - minimization
   - perspective

An example is $f(x) = ||Ax + b||$.

The function $\lambda_{max}(X) : \mathbf{S}^n \to \mathbb{R}$ is a convex function on the space of symmetric matrices.

Suppose $X = X^T$. Let's consider $\max y^T X y$ such that $y^T y = 1$. The constraint qualification holds because the gradient is $2y$: $\{\nabla g_i(y^*)\}$ is linearly independent. So, the Lagrangian here is:

$$L(y, \lambda) = y^T X y + \lambda(1 - y^T y)$$

Then, we set equal to zero:

$$0 = \nabla L(y, \lambda) = Xy - \lambda y = 0$$

So, $Xy = \lambda y$, and $y^T X y = \lambda y^T y = \lambda$, and we're trying to maximize this. Using this kind of technique, how do we find the second-largest eigenvalue? We know from linear algebra that we have orthogonal complements for symmetric matrices. Thus, we can jut add in the constraint $y_1^T y = 0$, where y_1 was optimal for the previous problem. Now, we're going to modify L for this problem as:

$$L(y, \lambda) = y^T X y + \lambda(1 - y^T y) + \mu y_1^T y$$

When we differentiate, we get

$$0 = \nabla L(y, \lambda) + \mu y_1$$

If we multiply through by y_1^T, we get

$$y_1^T X y - \lambda y_1^T y + \mu y_1^T y_1$$

Now, the second term cancels from orthogonality, and the first term cancels because of the eigenvalue. So, we have one remaining term, and we get that $\mu = 0$. So, again, we get

$$Xy = \lambda y.$$

This is a nice proof of the spectral theorem. Of course, we're using the power of optimization and Lagrange multipliers, but this is usually a complicated theorem in linear algebra too.

The smallest eigenvalue $\lambda_{min}(X)$ is a concave function, and we can efficiently maximize this. If you want to build a bridge, you have to try to guarantee that all the real parts of all the eigenvalues are bounded away from zero. So, you want to push away the maximum real part for the smallest eigenvalue to be far away from zero. So, our λ_{max} is important, and comes up in many applications.

6.14 Compositions of functions

The composition $h \circ g$ is convex under certain compositions.

There are similar statements for vector functions.

6.15 Minimization

If $f(x, y)$ is convex in (x, y) and C is a convex set, then

$$g(x) = \inf_{y \in C} f(x, y)$$

is convex.

Thus, if we have a linear program $P(\epsilon)$ = minimize $c^T x$ such that $Ax = b + \epsilon, x \geq 0$, we can apply this.

One example: $f(x, y) = x^T A x + 2x6TBy + x^T C y$ with

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0, C \succ 0$$

$g(x) = \inf_y f(x,y)$ is convex, so ...

The distance from a point x to a set S

$$\mathbf{dist}(x,S) = \inf_{y \in S} ||x-y||$$

is convex if S is convex.

The conjugate

$$f^*(y) = \sup_{x \in dom(f)} (f^T x - f(x))$$

is function, even if f is not. This leads to the theory of Fenschel duality (as opposed to Lagrangian duality).

The *quasiconvex* functions are the ones where the sublevel sets are convex. These and *pseudoconvex* functions come up a lot in economics.

There is a modified Jensen inequality for quasiconvexity. There are first-order conditions for quasiconvexity.

6.16 Log-concave and log-convex functions

A positive function f is log-concave if $\log f$ is concave:

$$f(\theta x + (1-\theta)y) \geq f(x)^\theta f(y)^{1-\theta} \text{ for } 0 \leq \theta \leq 1$$

f is log-convex if $\log f$ is convex.

The function $-\log\det(X)$ is strictly concave $X \succ 0$. This function goes to $+\infty$ as X becomes singular (that is, goes to the boundary of positive definite matrices).

6.17 K-convexity

Example $f : \mathbf{S}^m \to \mathbf{S}^m$ given by $X \mapsto X^2$ is $\mathbf{S}^m_+$-convex (on $\mathbf{S^m}$).

Exercise: Show $X^T X - I$ ($X \in \mathbb{R}^{n \times n}$) is K-convex.

We say that $G(X)$ is K-covex iff $\langle \varphi, G(X) \rangle$ is convex for all $\varphi \in K$.

7 Optimization problems

7.1 Standard form

An optimization problem in standard form is to minimize $f_0(x)$ subject to $f_i(x) \leq 0$, $(i = 1, \ldots, m)$ and $h_i(x) = 0$, $(i = 1, \ldots, p)$.

We should be clear that by minimum, we really mean infimum: the greatest lower bound! Mainly, we will be interested in finding *locally optimal* solutions.

We have a typical example in $f_0(x) = x^3 - 3x$. Then, $p^* = -\infty$, but we have a local optimum at $x = 1$. This is a local optimum by looking at the derivative and second derivative of f at the point $x = 1$. Local to $x = 1$, we have a strictly convex function.

The standard form has an *implicit constraint*. We take the intersection of all the domains of the functions f_i and h_i.

We can study the feasibility problem by trying to minimize a constant function.

7.2 Convex optimization problem

A convex optimization problem: It's in standard form where the f_i are convex and the equality constraints are all affine.

More generally, the problem is *quasiconvex* if the f_0 is quasiconvex and f_i (for $i > 0$) are convex.

For convex problems, a solution is a local optimum **if and only if** it is a global optimum. It is possible to have multiple global optima: We can just picture a function that is flat for a while.

7.3 Optimality criterion for differentiable f_0

x is optimal if and only if it is feasible and

$$\nabla f_0(x)^T(y - x) \geq 0$$

for all feasible y.

Theorem 7.1. *(Fermat's Theorem)* $\overline{x} \in argmin(f(x)) \implies \nabla f(\overline{x}) = 0$.

Proof. If $\nabla f(\overline{x}) \neq 0$, then set $d = -\nabla f(\overline{x})$. Then $f(\overline{x}+\alpha d) \sim f(\overline{x})+\alpha\nabla(f\overline{x})^T d$.

Now, for $\overline{x} \in X$. Suppose that $d = -\nabla f(\overline{x})$. Again, if we were to move a little bit again, we'd have a contradiction. If we can take any direction from $\overline{x}$, we could end up inside the set.

$\nabla f(\overline{x})^T d \geq 0$ for all $d \in X - \overline{x}$. That is, $\nabla f(\overline{x}) \in (X - \overline{x})^+$. □

This is called Rockaffelar-Pshenichnyi.

So, this turns into a very nice optimality characterization when minimizing a convex function.

If we are minimizing $f(x)$ over $x \in \mathbb{R}^n_+$, that is, say f is convex. Then,

$$\overline{x} \in arg \min_{x \in \mathbb{R}^n_+} f(x)$$

if and only if $\nabla f(\overline{x}) \in (\mathbb{R}^n_+ - \overline{x})^+$.

7.4 Modifying the problem

All inequalities can be converted to equality constraints (with slack variables). One can introduce equality constraints.

Another useful thing: Instead of minimizing $f(x)$, you minimize t subject to $f(x) \leq t$. A lot of times you'll see this little trick being done.

7.5 Piecewise linear minimization

This is a generalization of linear programming where your objective is to minimize the maximum of a finite collection of affine linear functionals.

This is equivalent to an LP.

One can find the *Chebyshev center* of a polyhedron by a linear program.

7.6 Generalized Linear Fractional Program

The objective function is a rational linear function.

7.7 Quadratic Programming (QP)

7.8 Quadratically constrained quadratic program (QCQP)

7.9 Second-order Cone Programming

These are more general than QCQPs.

7.10 Robust Linear Programming

There may be uncertainty (say in an LP) in the c_i, b_i, a_{ij}. If you want to guarantee that things don't go wrong, there are two common approaches to handling uncertainty.

One can have a deterministic model, where the constraints (that we state) must hold for all $a_i \in \mathcal{E}_i$, that is, for all possible values for a_i. You will have an infinite number of constraints if $\mathcal{E}_i$ is infinite.

Another approach is probabilistic.

7.11 Geometric Programming

A *posynomial* is a sum of monomials

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}}$$

7.12 Application: Minimizing the spectral radius of a nonnegative matrix

Theorem 7.2. *(The Perron-Frobenius Theorem)*

This is how Google started. We'll use a Markov matrix (a special type of positive matrix). We'll study the random processes and see this Markov matrix. Then, it finds the eigenvector corresponding to this positive matrix. And this vector gives you the rankings of the positive nodes. There's an article by Clete Moler.

Every month, Google would solve the eigenvector for the largest eigenvalue for a very large matrix. The largest eigenvalue was found by the power method.

Proposition 7.3. *(Vandergraff, Elsner, ...) Let K be a closed convex cone. Let A be a linear operator such that $A(K) \subset int(K)$. Then there is an eigenvector $v \in K$.*

7.13 LP and SOCP as SDP

A linear program **is** a semi-definite program. The problem:
(LP) minimize $c^T x$ subject to $Ax \leq b$ can be rewritten as
Minimize $c^T x$ subject to $\mathbf{diag}(Ax - b) \leq 0$.

7.14 Eigenvalue minimization

Suppose we wish to minimize $\lambda_{max}(A(x))$ where $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$, where the A_i are symmetric $k \times k$ matrices. We reformulate this as an equivalent SDP:

Minimize t subject to $A(x) \preceq tI$ where the variables are $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$. This follows from

$$\lambda_{max}(A) \leq t \iff A \preceq tI.$$

7.15 Matrix norm minimization

If we want to minimize $||A(x)||_2$ where $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$, where the A_i are symmetric $p \times q$ matrices.

We can reformulate this as: Minimize t subject to

$$\begin{bmatrix} tI & A(x) \\ A(x)^T & tI \end{bmatrix} \succeq 0$$

The variables here are $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$. The constraint follows from

$$||A||_2 \leq t \iff A^T A \preceq t^2 I,\ t \geq 0 \iff \begin{bmatrix} tI & A(x) \\ A(x)^T & tI \end{bmatrix} \succeq 0.$$

7.16 Regularized Least-Squares

Instead of minimizing $||Ax - b||_2^2$, minimize $(||Ax - b||_2^2, ||x||_2^2) \in \mathbb{R}_+^2$.
Aside on CAT scans, and accuracy.

8 Duality

We need to understand the general duality theory before looking at interior point methods. We'll see how to derive an interior point algorithm based on duality. Maybe you can figure out how to develop one for your problem in a similar way. For simplicity, let's look at the convex case.

Suppose we have a program:

$$\begin{array}{rc} min & f(x) \\ s.t. & g_k(x) \leq 0, k = 1, \ldots, m \\ & f, g_k \text{ are convex on } \mathbb{R}^n \end{array}$$

The level sets are convex, so the feasible set $\mathcal{F} = \{x \mid g_k(x) \leq 0 \ \forall \ k\}$ is a convex set. So $\overline{x}$ is optimal if and only if $\nabla f(\overline{x}) \in (\mathcal{F} - \overline{x})^+$. So this works all the time: it's a very geometric approach. If we want to do things computationally, we have to use gradients.

So, we have the algebraic formulation of this geometric idea:

$$\nabla f(\overline{x}) = - \sum_{k \in A(\overline{x})} \lambda_k \nabla g_k(\overline{x}), \qquad \lambda_k \geq 0$$

$$\nabla f(\overline{x}) + \sum_{k \in A(\overline{x})} \lambda_k \nabla g_k(\overline{x}) = 0$$

$$\nabla f(\overline{x}) + \sum_k \lambda_k \nabla g_k(\overline{x}) = 0 \text{ and } \lambda_k g_k(\overline{x}) = 0 \ \forall \ k$$

This last condition $\lambda_k g_k(\overline{x}) = 0 \forall k$ is called complementary slackness: at least one of these has to be zero. Here we see the idea of *active constraints*.

We examine a case where the gradients of g_1 and g_2 do not match. Here, the algebra and geometry do not match. To avoid this, we need a *constraint qualification* (CQ). For convex programs, we will use Slater's condition (strict feasibility).

Slater's Condition: There exists an $\hat{x}$ such that each ? in $\mathcal{F}$ satisfies strict feasibility on all constraints. But we can do this more generally:

Suppose we have a convex problem, and assume that p^* is finite:

$$p^* = \inf\{f(x) : G(x) \preceq_K 0, \ x \in \Omega\}$$

where $f(x)$ is convex on Ω, G is K-convex on Ω, K is a proper closed convex cone, and Ω is a convex set.

Slater's constraint qualification for this is:

$$\exists \ \hat{x} \in \Omega \text{ such that } G(\hat{x}) \prec_K 0$$

Then, we can say that under this condition, $\exists \ \lambda^* \in K^+$ such that

$$p^* = \inf_{x \in \Omega} f(x) + \langle \lambda^*, G(x) \rangle$$

This is strong duality (the existence of λ^*). If this inf is attained at x^* feasible, then

$$\langle \lambda^*, G(x^*) \rangle = 0 \quad \text{(complementary slackness)}$$

All we've talked about are necessary conditions (necessary conditions are what give us algorithms).

Necessity requires the constraint qualification. Sufficiency holds without the constraint qualification (in the convex case). I.e., if

$$\exists\ \lambda^* \succeq_{K^+} 0 \tag{2}$$

and

$$\begin{aligned} d^* &= \inf_{x \in \Omega} L(x, \lambda^*) \\ &= F(\overline{x}) + \langle \lambda^*, G(\overline{x}) \rangle, \quad \overline{x} \text{ feasible} \end{aligned}$$

and $\langle \lambda^*, G(\overline{x}) = 0$, then $\overline{x}$ is optimal. So we have dual feasibility (condition (2) and $\nabla L(\overline{x}, \lambda^*) \in (\Omega - \overline{x})^+$).

So, we have primal feasibility and dual feasibility.

So, say we have a problem, and we have

$$L(x, \lambda, 0)$$

We have the *Karush*[3]*-Kuhn-Tucker* optimality conditions (these are necessary). Need a CQ.

Suppose we have $\overline{x}$ a feasible solution, and we want to see if it's optimal. The most famous condition in this area is called the *Mangasarian-Fromovitz*[4] CQ. This condition says that at $\overline{x}$, $\exists\ d$ such that $\nabla f_i(\overline{x})^T < 0$ for all i with $f_i(\overline{x}) = 0$ and also we have $\nabla h_i(\overline{x})^T d = 0$ for all $i \in A(\overline{x})$. Finally, we also have $\{\nabla h_i(\overline{x} : \forall i\}$ must be linearly independent. If you remember from calculus, the Lagrange multipliers fail if you don't have linear independence. Our small example earlier shows this.

8.1 Linear Programming

Let's apply this to linear programming and develop an interior point method. Let's look at the instance LP.

The primal program is

$$\begin{aligned} \mathit{min} \quad & c^T x \\ \mathit{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

and the dual is

$$\begin{aligned} \mathit{max} \quad & b^T y \\ \mathit{s.t.} \quad & A^T y \leq c \\ & A^T y + z = c \\ & z \geq 0 \end{aligned}$$

[3] A masters student at the University of Chicago who wrote about this in his thesis. He did this before 1945, probably somewhere around 1939.

[4] I visited the University of Maryland for a semester, and I saw the name Fromovitz on the door. When I asked him if he was the one whose name is on the CQ, he said, "Yes, I am, but don't ask me about it."

We all know Dantzig's simplex method (1948).

What are the optimality conditions within our framework? We'll always have $x \geq 0, z \geq 0$. Let's look at dual feasibility first: $A^T y + z - c = 0$. Then we also have primal feasibility: $Ax - b = 0$. For the complementary slackness, I'm going to write

$$ZXe = 0$$

where $Z = Diag(z)$, $X = Diag(x)$, and e is the vector of all ones.

The primal simplex method maintains primal feasibility and complementary slackness. It's moving from vertex to vertex until you get to dual feasibility. This is the data in the bottom row of your tableau. The dual simplex method works by starting with dual feasibility and complementary slackness.

In the 1960s, Klee and Minty came up with a polytope that takes exponential time. Khachian then came up with a polynomial time (ellipsoid method) algorithm (1979) for linear programming, but the algorithm is very inefficient in practice. The simplex method (on average) solves problems in linear time (Schneil (sp?)). The Hirsch Conjecture is a conjectured bound on the diameter of a polyhedron. Karmarkar developed an interior point method that was much faster[5].

8.2 Log-barrier Problem

We will apply log-barrier to (D). So we have

$$b^T y + x^T(c - A^T y - z)$$

So instead of solving (D), we'll look at this. We'll use a barrier to capture the idea of $z \geq 0$. So, we have: Maximize:

$$B_\mu(x, y, z) = b^T y + x^T(c - A^T y - z) + \mu \sum \log z_i \tag{3}$$

If we differentiate $\frac{d}{dx}$, we have

$$c - A^T y - z = 0$$

the statement of dual feasibility. If we differentiate with respect to y, we get

$$b - Ax = 0$$

and out pops primal feasibility. If we differentiate with respect to z, we get

$$X - \mu Z^{-1} = 0$$

If we write this out slowly,

$$\frac{d}{dz_i} = -x_i + \mu \frac{1}{z_i} = 0 \quad (\mu > 0) \tag{4}$$

[5] Karmarkar got a job with AT&T. He claimed that he could beat the simplex method by a constant factor. Like for Khachian, there was a big big fuss. He signed a non-disclosure agreement, so he couldn't tell anybody how the algorithm worked.

So, we bring the x_i over and move the z_i out: Thus, $\mu - x_i z_i \forall i$. This was originaally never tried on linear programming. Now, we send μ down to zero, and we'll get closer and closer to the boundary. So, as μ goes to zero, we are getting our complementary slackness. So, we are converging to the optimal.

You want to solve this with Newton's method, but the inverse matrix is highly non-linear. So, we change this last line by multiplying through by Z, and then we might as well multiply by e, and so we can rewrite (4) as

$$-ZXe + \mu e = 0.$$

Each time we solve these equations for $\mu > 0$, we're solving the problem (3). The *central path* (a path that is a function of μ) is the solutions to the problem (3) for each μ. The solution is a unique point (x_μ, y_μ, z_μ) in primal-dual space $\{(x, y, z)\}$.

One can ask, why can't you just set μ equal to zero right away? There are some approaches of this kind. But the basic problem (for us at this point) is that we might not have convergence, because there are other stationary points that you can converge to. So, the idea is to stay within a neighborhood of the central path (since actually staying in the path might be too much work). You reduce μ and you stay in these region, and you converge to optimality. This is called the central path method.

There's still many questions: How do you solve each of these problems? How do you do the iteration? How do you decrease μ? (You might even increase μ. It's a dynamic process.)

You could start, say, at x and z are all ones.

The central path is analytic (smooth), but it's not necessarily

Theorem 8.1. *(Karush-Kuhn-Tucker) Let $\overline{x}$ be optimal, and suppose that the Mangasarian-Fromovitz conditions hold at $\overline{x}$. Then $\exists\ \lambda^* \geq 0, \nu^*$ such that $0 = \nabla L(\overline{x}, \lambda^*, \nu^*),\quad \lambda_i^* f_i(\overline{x}) = 0\ \forall\ i$.*

Back to the LP. Let's see how we develop an algorithm for this. Let's write down the equations. Let

$$F_\mu(x, y, z) = \begin{pmatrix} A^T y + z - c \\ Ax - b \\ ZXe - \mu e \end{pmatrix} = 0$$

Given current estimates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \equiv S, \quad x > 0, z > 0$$

How should we choose the parameter μ? This is a good time to choose μ. We want $ZXe = \mu e$, so if we just take the transpose on both sides, $\Longrightarrow$ $e^T ZXe = \mu e^T e = n\mu$. What is $e^T ZXe$? This is just $z^T x = n\mu$, so this is just $\mu = \frac{z^T x}{n}$.

Recall that if x and z are feasible, then the numerator here $z^T x$ is our duality gap. So, we want μ to be $\frac{1}{n}$ times to duality gap. So this shows us in a different way that decreasing μ leads to a convergence to optimality. But, we want to be more aggressive in choosing μ. In practice, $\mu = \frac{z^T x}{10n}$. In SDP, the 10 is replaced by a 2. These things are just observations through numerical evidence.

Now, we have these equations. How do we solve these equations? With Newton's Method!

We want to use Newton's method to solve $F_\mu(S) = 0$. This is based on a linearization. We simply have

$$F_\mu(S) \approx F_\mu(S) + F'(S)(dS) = 0$$

where $F'(S)$ is the Jacobian. We solve the Newton equation:

$$F'_\mu(s)dS = -F_\mu(S)$$

So, let's recall what happens when we learned it in calculus (geometrically). We can pictorally see how quickly this method is going to converge. Similarly, you can see how quickly you can diverge if you start at a bad initial point. So, we need to differentiate this function $F_\mu(S)$. We're going to differentiation with respect to x, y, and z. The respective derivatives are

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix}$$

So, we have

$$F'_\mu(S) \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = -F_\mu(S) = \begin{pmatrix} -r_d \\ -r_p \\ -r_c \end{pmatrix}$$

One of the interesting things about this Jacobian is all these zeroes. We can exploit this by using **block** Gaussian elimination. This is very stable and very fast (with a Cholesky factorization).

To make things a little bit easier for ourselves, let's just assume that we have primal/dual feasible.

In our block elimination, eliminate dz, then dx, then isolate dy. Our first step is to isolate dz, by using equation 1. For simplicity, we assume that $r_d = r_p = 0$, so we have a feasible start. (The great thing about Newton's method is that you always retain the linear relations, even if you don't go a full step.) So, let's use equation 1. Because r_d is zero, we have

$$dz = -A^T \ dy$$

so now I've eliminated equation number 1. I want to substitute this into equation 3. Equation 3 has

$$Z \ dx - XA^T \ dy = -ZXe + \mu e$$

The advantage of starting interior and staying interior is that the inverse always exists, so we left-multiply by it:

$$dx = Z^{-1}XA^T\ dy - Xe + \mu Z^{-1}e$$

So now we substitute into equation 2. Here, we get

$$A(Z^{-1}XA^T\ dy - Xe + \mu Z^{-1}e) = 0.$$

By expanding this, we have:

$$AZ^{-1}XA^T\ dy = b - \mu AZ^{-1}e$$

So $AZ^{-1}XA^T$ is an $m \times m$ square matrix, and our right hand side vector is $b - \mu AZ^{-1}e$. So, we solve for dy in this linear system. Note that $AZ^{-1}XA^T \succeq 0$. Why? $S = Z^{-1}X$ is diagonal and positive, so we can take a square root:

$$AS^{\frac{1}{2}}S^{\frac{1}{2}}A^T = (AS^{\frac{1}{2}})(AS^{\frac{1}{2}})^T$$

and this is always $\succeq$. It is $\succ$ iff $AS^{\frac{1}{2}}$ is full row rank. In CPLEX, they check this and get rid of redundant rows. You never start an LP until you have full row rank. (A book by Gilbert Strang shows that matrices of this kind come up in mathematical physics.) It's positive definite, so we solve this using “sparse” Cholesky factorization:

$$P = LL^T$$

(In the above, replace μ with τ, and set $\tau = \sigma\mu$.)

If we have $\tau = \sigma\mu$, and $F_\tau(x, y, z) = 0$, then setting $\sigma = 1$, you would move closer to the central path, but not improving (significantly) your objective distance. If you set $\sigma = 0$, that would be the affine scaling direction, and that's aiming for the optimal direction right away. But if you're too far away, it's too hard to get a good aim at. So, $\sigma \in [0, 1]$. So, in these problems, you'll modify σ based on the type of problem you have.

If μ decreases a lot, you just made a really good step. Sometimes in an SDP, you can take $\sigma > 1$. Then, you're concerned about centering yourself before improving any more. (Factorization is a big area, and you can find a lot of webinars.)

So the algorithm. We have an algorithm:

Algorithm

1. (a) $x_0 > 0, z_0 > 0$. Initiate $\mu = \frac{x_0^T z_0}{n}$. Then set your τ (as above, say).

   (b) Find Newton direction, i.e., find dy using the "normal equations."

   (c) Back-solve for dx and dz.

2. Take a step to the boundary. (If you want polynomial time, you go all the way to the boundary and backtrack just a little bit.) But, let's just take care of the algorithm is practice (this is what OB1 does). We do

   $$x_0 + \alpha_{max}\ dx \geq 0$$

   Find the biggest α. Then, we back-track to stay interior (by multiplying by 0.9997.) For example, let $\alpha = \min\{0.9997 \cdot \alpha_{max}, 1\}$. Then assign

   $$x_1 = x_0 + \alpha\ dx$$

   Similarly, for z,

   $$z_1 = z_0 + \beta\ dz$$

   where $\beta = \min\{0.9997 \cdot \beta_{max}, 1\}$, and finally,

   $$y_1 = y_0 + \beta\ dy$$

   (We choose the same scaling β to maintain feasibility.)

3. Repeat (at step 1).

The only thing now is just deciding when to stop. We need a criterion (for what we will consider optimality). Typically, we can only get about 8 decimals of accuracy. In the above, the Z^{-1} is done in the code. As you can see, this should be very easy to code up!!

[The next thing to do: What happens to semi-definite programming? We should look at the Gauss-Newton method, not the Newton method.] Here, we have ZX was a diagonal times a diagonal. In SDP, if Z and X are symmetric, ZX might note be, so we have a map

$$\mathbf{S}^n \times \mathbf{S}^n \to GL^n.$$

We have to figure out what we're going to do about that.

Define

$$F(\lambda) = \mathcal{A}(A^*\lambda)_+ - b = 0$$

In general, this is not differentiable, but in finite dimension, you can do everything: each piece of the composition is differentiable and you have the chain rule.

Given the Lagrange function $L(x, \lambda, \nu)$, we have the Lagrangian dual funciton

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu)$$

We can consider the **minimization** version of the max-cut problem, and we call this *two-way partitioning.*

$$\begin{array}{cc} min & x^T W x \\ subj. & x_i^2 = 1, \quad i = 1, \dots, n \end{array}$$

Recall the SDP relaxation of max-cut was

$$\begin{array}{cc} max & x^T L x \\ subj. & x_i^2 = 1, \quad i = 1, \dots, n \end{array}$$

We could also write $x^T L x = tr(x^T L x) tr(L x x^T)$. So the relaxation would be to relax the rank $= 1$ constraint. So, instead, we maximize $tr(LX)$ subject to the constraint

$$diag(X) = e, X \succeq 0$$

If we set

$$p^* = \max tr(LX) \text{ s.t. } \text{diag}(X) = e, X \succeq 0$$

then we can compute

$$d^* = \min_y \max_{X \succeq 0} tr(LX) + y^T (e - diag(X)$$

As with LP, we just move things around. We have

$$d^* = \min_y \max_{X \succeq 0} \langle L, X \rangle - \langle y, diag(X) \rangle + y^T e$$

$$d^* = \min_y \max_{X \succeq 0} \langle L, X \rangle - \langle Diag(y), X \rangle + y^T e$$

The adjoint for Diag is diag:

$$diag = Diag^*$$

So,

$$d^* = \min_y \max_{X \succeq 0} \langle L - Diag(y), X \rangle + y^T e$$

So the hidden constraint is that

$$L - Diag(y) \preceq 0$$

Now, we have a primal-dual pair: With the hidden constraint, the optimum is at $X = 0$, so the max and the first term disappears, and the dual problem (D) is:

$$\begin{array}{cc} min & y^T e \\ s.t. & Diag(y) \succeq L \\ & Diag(y) - Z = L \\ & Z \succeq 0 \end{array}$$

The statement of compentary slackness for (P) and (D) is

$$\langle Z, X\rangle 0 \iff ZX = 0$$

$$\ddot\smile$$

We are looking for an X, Z, Y that satisfies these conditions, and then we will have optimality:

$$\begin{aligned} Diag(y) - Z - L &= 0, Z \succeq 0 && (5)\\ diag(X) - e &= 0, X \succeq 0 && (6)\\ XZ &= 0 && (7) \end{aligned}$$

What is the derivative of $\mu \log\det X$? It's μX^{-1}.

So, we can differentiate

$$B_\mu(X, y) = \max_X \langle X, L\rangle + y^T(-diag(X) + e) + \mu \log\det X$$

to get

$$\nabla_x B_\mu(X, y) = L - Diag(y) + \mu X^{-1} = 0$$

and

$$\nabla_y B_\mu(X, y) = e - diag(X) = 0$$

We can define

$$Z = \mu X^{-1}$$

to make these equations look like the three equations (5), (6), and (7).

So we want to differentiate $F_\mu(X, y, Z) = 0$ in order to find a Newton direction.

So look at

$$F'_\mu(X, y, Z) \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} = -F_\mu(X, y, Z)$$

Computing F'_μ,

$$\begin{bmatrix} & & Diag(\Delta y) - \Delta Z \\ diag(\Delta X) & & \\ Z\Delta X & +\Delta Z X & \end{bmatrix} = \begin{bmatrix} -R_d \\ -r_p \\ -R_c \end{bmatrix}$$

So we have

$$\begin{bmatrix} 0 & Diag(\cdot) & -I(\cdot) \\ diag(\cdot) & 0 & 0 \\ Z(\cdot) & 0 & (\cdot)X \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} = \text{RHS}$$

MatLAB will use a Cholesky decomposition on a symmetric matrix, even if you didn't ask for it.

8.3 Interior point methods on the SDP relaxation of the Max-Cut problem

```
function [phi, X, y] = maxcut( L);
% solves: max trace(LX) s.t. X psd, diag(X) = b;  b = ones(n,1)/4
%         min b'y       s.t. Diag(y) - L psd, y unconstrained,
% input:  L ... symmetric matrix
% output: phi ... optimal value of primal, phi =trace(LX)
%         X   ... optimal primal matrix
%         y   ... optimal dual vector
% call:   [phi, X, y] = psd_ip( L);

digits = 6;                     % 6 significant digits of phi
[n, n1] = size( L);             % problem size
b = ones( n,1 ) / 4;            % any b>0 works just as well
X = diag( b);                   % initial primal matrix is pos. def.
y = sum( abs( L))' * 1.1;       % initial y is chosen so that
Z = diag( y) - L;               % initial dual slack Z is pos. def.
phi = b'*y;                     % initial dual
psi = L(:)' * X( :);            % and primal costs
mu = Z( :)' * X( :)/( 2*n);     % initial complementarity
iter=0;                         % iteration count

disp(['      iter    alphap    alphad      gap     lower      upper']);

while phi-psi > max([1,abs(phi)]) * 10^(-digits)

      iter = iter + 1;          % start a new iteration
      Zi = inv( Z);             % inv(Z) is needed explicitly
      Zi = (Zi + Zi')/2;
      dy =  (Zi.*X) \ (mu * diag(Zi) - b);      % solve for dy
      dX = - Zi * diag( dy) * X + mu * Zi - X;  % back substitute for dX
      dX = ( dX + dX')/2;       % symmetrize
% line search on primal
      alphap = 2;               % initial steplength changed to 2
%     alphap = 1;               % initial steplength
      [dummy,posdef] = chol( X + alphap * dX );  % test if pos.def
      while posdef > 0,
              alphap = alphap * .8;
              [dummy,posdef] = chol( X + alphap * dX );
              end;
      alphap = alphap * .95;  % stay away from boundary
% line search on dual; dZ is handled implicitly: dZ = diag( dy);
%     alphad = 1;
      alphad = 2;               % initial steplength changed to 2
      [dummy,posdef] = chol( Z + alphad * diag(dy) );
      while posdef > 0;
              alphad = alphad * .8;
              [dummy,posdef] = chol( Z + alphad * diag(dy) );
              end;
      alphad = alphad * .95;
% update
      X = X + alphap * dX;
      y = y + alphad * dy;
      Z = Z + alphad * diag(dy);
      mu = X( :)' * Z( :) / (2*n);
      if alphap + alphad > 1.8, mu = mu/2; end; % speed up for long steps
      phi = b' * y; psi = L( :)' * X( :);
% display current iteration
        disp([ iter alphap alphad (phi-psi) psi  phi ]);

        end;            % end of main loop
```

9 Summary

Let's run through quickly a small review of what we've done last week. The main theme has been convexity. We looked at things related to convex sets. In particular, we studied convex cones. One of the important results we looked at was the hyperplane separation theorem (and also the support theorem). One of the interesting lemmas that we got from this was

$$K = K^{++} \text{ if and only if } K \text{ is a closed convex cone.}$$

Then, we went into convex functions. We looked at a generalized convex function

$$G(\lambda x + (1-\lambda)y) \preceq_K \lambda G(x) + (1-\lambda)G(y) \ \forall\ 0 \leq \lambda \leq 1\ \forall\ x, y \text{ in domain}$$

For example, we had the special case $K = \mathbb{R}^m_+$ ($G : \mathbb{R}^n \to \mathbb{R}^m$). We had that $f : \mathbb{R}^m \to \mathbb{R}$ is a convex on Ω (convex) if $\nabla f(x)^T(y-x) \leq f(y) - f(x)\ \forall\ x, y \in \Omega$. This formula it talking about the supporting tangent hyperplanes on the graph of f. We had the zeroth-order and first-order conditions. The second-order condition is: $f : \mathbb{R}^n \to \mathbb{R}$ is convex on Ω if $\nabla^2 f(x) \succeq 0\ \forall x \in \Omega$. If $f : \mathbb{R}^n \to \mathbb{R}$ is convex, Ω is a convex set, $\overline{x} \in argmin_{x \in \Omega} f(x)$ iff $\nabla f(\overline{x}) \in (\Omega - \overline{x})^+$.

Let $T = T(\Omega, \overline{x})$ denote the tangent cone of Ω at $\overline{x}$. T consists of those directions d one can travel in from $\overline{x}$. $d = \lim d_k$. This second-order statement basically says that the inner product against any feasible direction has to be greater than zero.

As an aside, this can be extended for the non-smooth function case as well. For convex functions, there's a nice smooth calculus. We can talk about the subgradient:

$$\partial f(\overline{x}) = \{\varphi \in \mathbb{R}^n : \varphi^T(y - \overline{x}) \leq f(y) - f(\overline{x}),\ \forall\ y\}$$

The pictorial idea is the same: instead of having one gradient, you have multiple gradients. For example, if $f(x) = |x|$, we know what the derivative of f at $x = 1$ is (say), so $\partial f(1) = \{1\}$ and $\partial f(0) = [-1, 1]$. We still have tangent cones for nonconvex sets S at $\overline{x}$. We look at $d_k = x_k\overline{x}$, $x_k \in F$. Define

$$d = \lim_{x_k \neq \overline{x}} \frac{x_k - \overline{x}}{||x_k - \overline{x}||},\ x_k \in F$$

Then $\overline{x} \in argmin(f(x))$, where f is a non-convex set. Then, we get the forward direction

$$\implies \nabla f(\overline{x}) \in (T(S, \overline{x}))^T$$

The next thing that we did was convex programing. We looked at

$$\begin{aligned} p^* = min &\quad f(x) \\ s.t. &\quad G(x) \preceq_K 0 \\ &\quad x \in \Omega \end{aligned}$$

where f is convex on Ω, K is a proper closed convex cone, Ω is a convex set, and G is K-convex on Ω. We used the Lagrangian

$$L(x, \lambda) = f(x)\langle\lambda, G(x)\rangle, \quad \lambda \succeq_{K^+} 0$$

Why? Because of the dual functional

$$\varphi(\lambda) = \min_{x\in\Omega} L(x, \lambda), \quad \forall\lambda \succeq_{K^+} 0$$

This minimum will always be a lower bound for our problem:

$$p^* \geq \varphi(\lambda)$$

So, now we get our dual problem is:

$$\max_{\lambda \succeq_{K^+} 0} \quad \varphi(\lambda)$$

We can also write this with the hidden constraints explicitly written in.

$$g(\hat{x}) \prec_K 0 \text{ for some } \hat{x} \in K \tag{8}$$

We have some theorems in this are: If P^* is finite and Slater's Constraints Qualification (equation (8)) holds, then

$$p^* = \varphi(\lambda^*) \text{ for some } \lambda^* \succeq_{K^+} 0$$

This is our necessary condition. So, this gives us something useful for an algorithm. In the convex case, we have sufficiency as well: if $\varphi(\lambda^*) = L(\overline{x}, \lambda^*)$ and $\overline{x}$ is feasible, then $\langle\lambda^*, G(\overline{x}) = 0$.

A primal-feasible and complementary slackness proveable sufficient condition for optimality:

$$\nabla L(\overline{x}, \lambda^*) \in (\Omega - \overline{x})^+, \quad \lambda^* \succeq_{K^+} 0 \text{ (Dual feasibility)}$$

$$\overline{x} \in \Omega, \quad G(\overline{x}) \preceq_K 0 \text{ (Primal feasibility)}$$

$$\langle \lambda^*, G(\overline{x}) = 0 \text{ (complementary slackness)}$$

So $\overline{x}, \lambda^*)$ is an optimal pair if and only if these conditions hold.

If $\overline{x}$ is feasible and λ^* is feasible,

$$\begin{aligned} f(\overline{x}) &\geq f(\overline{x}) + \langle \lambda^*, G(\overline{x}) \rangle \\ &\geq \min_{x \in \Omega} L(x, \lambda^*) = \varphi(\lambda^*) \end{aligned}$$

In LP, $\lambda^T(b - AX) = \sum \lambda_i (b - Ax)_i = 0$ (note, $\lambda_i \geq 0$ and $(b - Ax)_i \geq 0$. These KKT conditions always have a physical interpretation: Remember the example of the diet problem and the pill salesman. These other prices are called *shadow prices.* If you have a positive shadow price, you're using all of that resource. These shadow prices tell you how your objective value will change with respect to changes in your constraints. You can look at

$$\frac{\partial p^*}{\partial b_i} \approx \lambda_i \tag{9}$$

So, now that you've modeled and solved your problem, you have to interpret your solution. So, more than just knowing the optimum, you want to know about what happens to equations of the form (9). Why? You want to be able to tell your boss what happened since that change between when you solved the problem and now.

Convexity takes this local condition and makes it a global condition.

We can finish off by looking at the non-convex case. Again, our problem is

$$\begin{aligned} min \quad & f(x) \\ s.t. \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

Here, we have $g(x) = (g_i(x)) : \mathbb{R}^n \to \mathbb{R}^m$ (for each active $i \in \mathcal{I}$), and $h(x) = (h_i(x)) : \mathbb{R}^n \to \mathbb{R}^p$ (for each active $i \in \mathcal{E}$).

At $\overline{x}$ feasible, we need a constraint qualification, such as the Mangararran-Fromowitz condition. We look at the active constraints $\mathcal{I}(\overline{x}) = \{i : g_i(\overline{x}) = 0\}$. Then the M-F CQ says

$$\exists \text{ a direction } d \ni \nabla g_i(\overline{x})^T d < 0 \; \forall \; i \in \mathcal{I}(\overline{x}), \; \nabla h_i(\overline{x})^T d = 0 \; \forall \; i \in \mathcal{E}$$

where the constraints $\{\nabla h_i(\overline{x})\}$ are linearly independent. (In the convex case, you can easily throw these away.) The M-F condition is equivalent to the generalized Slater CQ in the convex case.

If $\overline{x}$ is a local optimum, and the M-F CQ holds, then

$$0 = \nabla L(\overline{x}, \lambda^*, \mu^*), \lambda^* \geq 0, \mu^* \text{ free} \quad \text{(Dual feasibility)}$$

$$\begin{aligned} g(\overline{x} \leq 0, h(\overline{x}) = 0 \text{ (Primal feasibility)} \\ \lambda^{*T} g(\overline{x}) = 0 \text{ (Complementary slackness)} \end{aligned} \tag{10}$$

These are necessary, but not sufficient. And they need a CQ, so they don't always work. The interior point methods will do things we've seen: The $g(\overline{x}) \leq 0$ conditions will be turned into equations with a slack variable. For algorithms, this gets changed into $\lambda_i^* g_i(\overline{x}) = 0 \ \forall \ i$. The reason for this is for the central path. This gives you a nice central path, where as for (10), I don't see how.

For non-convex,

$$\min_{\lambda} \quad f(x)$$

We know $\overline{x} \in argmin(f(x))$ (or even local min) $\Longrightarrow \nabla f(\overline{x}) = 0$ and $\nabla^2 f(\overline{x}) \succeq 0$. And, if we go the other way, $\nabla f(\overline{x}) = 0, \nabla^2 f(\overline{x}) > 0 \Longrightarrow \overline{x} \in argmin(f(x))$.

Question: Can we know how many optimal values there are? Is this easy? *No. This is hard as finding zeroes of functions. Perhaps somebody who knows something about differential geometry might be able to say a word or two?*

If we had a bound on the Lipschitz constant, then we'd know what size our neighborhoods would have to be to see that our stationary point is focused on. This is what global optimization does: it takes bounds on the derivatives, and discretizes the region, and it will go finer and finer. At some point, it knows it's done, because there's no way to get more wiggle than this.

$$\begin{bmatrix} \nabla g_1 & \nabla g_2 & \cdots \end{bmatrix} \lambda = -\nabla f(\cdot) \equiv h \tag{11}$$

is inconsistent (for $\lambda \geq 0$) iff **(this was not finished on the chalkboard!!)**.

Equation (11) is a kind of Farkas' lemma. Recall:

$$A\lambda = b, \lambda \geq 0 \text{ is infeasible} \Leftrightarrow A^T d \leq 0, b^T d > 0 \text{ is feasible.}$$

(That is, d is a "good" search direction.) $\overline{x}$ is the current point. Then, you look at $x(t) = \overline{x} + td$ (Make improvement in the direction d, for small t).

If $d = -\nabla f(\overline{x}) \neq 0$, then $d^T \nabla f(\overline{x}) = -||\nabla f(\overline{x})|| < 0$.

If you have a non-convex problem, all kinds of crazy things are going to break out. There might be any feasible direction. **The extension from convex to non-convex is very hard. It's almost impossible.** But, here's a very special case. It's called a *trust region subproblem*:

$$\begin{array}{ll} min & q_0(x) = x^T A x + 2b^T x \\ s.t. & ||x|| \leq \delta \end{array}$$

If we like we can just square the last line. A is symmetric, but not necessarily PSD. The level sets for this could look like hyperbolas of two sheets. Now, you can ask: "Where's the minimum?" This problem is not convex, but its dual is (well, the dual always is convex), but there's a *ZERO duality gap*, that is, strong duality holds. So, this is a *hidden convex problem.* It sits on the boundary of convex and non-convex problems.

Let's look at this problem and the duality of this problem, and look at regularization.

$$A = \begin{bmatrix} -\nabla g_1 & -\nabla g_2 & \cdots & -\nabla g_k \end{bmatrix}, b = \nabla f(x_0)$$
$$b = A\lambda, \lambda \geq 0 \Leftrightarrow b \in cone\{\nabla g_k\}_{i=1}^{k}$$

K is a closed convex cone, so $K = K^{++}$. So $b \in K$ iff $d \in K^+ \implies b^T d \geq 0$ iff $\nabla g_i^T d \geq 0 \implies b^T d \geq 0$ iff $\nabla g^T d \leq 0 \implies \nabla f^T d \leq 0$.

For an LP, we'd have minimize $b^T d$ subject to $A^T d \geq c$.

$A\lambda = b, \lambda \geq 0$ is infeasible $\iff \exists\ d$ such that $\nabla_i g_i^T d \leq 0, \nabla f^T d < 0$.

So, we have our first constrained algorithm. We just use LP, e.g.,

$$\begin{array}{cc} min & \nabla f^T d \\ s.t. & \nabla g_i^T d \leq 0\ \forall\ i \\ & ||d||_\infty \leq 1\ (-1 \leq d_i \leq 1) \end{array}$$

But instead of minimize, we can say

$$\begin{array}{cc} z^* = \min & t \\ s.t. & \nabla f^T d \leq t \\ & \nabla g_i^T d \leq t\ \forall\ i \\ & ||d||_\infty \leq 1\ (-1 \leq d_i \leq 1) \end{array}$$

Under Slater's condition, you can prove that $z^* = 0$ iff the current point $\overline{x}$ is optimal. If your gradient is zero, you're done. If not, keep going. It gives you a search direction d. $z^* < 0$ iff d^* is a good search direction.

Let's go back to the trust region sub problem.

10 Non-convex objectives with strong duality

10.1 Trust Region Subproblem

This is an example of a quadratically constrained quadratic program (a QQP). Suppose you had

$$\begin{array}{cc} min & q_0(x) \\ s.t. & q_i(x) \leq 0, \quad i = 1, \ldots, m (i \in \mathcal{I}) \\ & (= 0 \text{ as well if desired}) \end{array}$$

And we should write each $q_i(x)$ in the form

$$q_i = \frac{1}{2} x^T Q_i x + b_i^T x + \alpha_i, \quad i = 0, 1, \ldots, m$$

The Lagrangian here is

$$L(x, \lambda) = q_0(x) + \sum_{i=1}^{m} \lambda_i q_i(x)$$

There are many problems that fall into this category. One example is the max-cut problem. We turned $x_i \in \{-1, 1\}$ to $x_i^2 = 1$. Similarly, we can take $x_i \in \{0, 1\}$ as $x_i^2 - x_i = 0$. It is powerful how semidefinite relaxation works for these. The Lagrangian dual will be

$$d^* = \max_{\lambda \geq 0} \min_x L(x, \lambda) = \frac{1}{2} x^T Q_0 x + b_0^T x + x^T \sum_{i=1}^{m} \lambda_i Q_i x + \sum \lambda_i b_i^T x + \sum \lambda_i \alpha_i$$

Now, there's a little trick that works. We add the constraint $x_0^2 = 1$.

$$d^* = \max_{\lambda \geq 0} \min_x L(x, \lambda) = \frac{1}{2} x^T Q_0 x + b_0^T x x_0 + x^T \sum_{i=1}^{m} \lambda_i Q_i x + \sum \lambda_i b_i^T x x_0 + \sum \lambda_i \alpha_i x_0^2$$

These really are the same. If $x_0 = -1$ in the optimal ,then x_0 is assigned 1 and x is assigned $-x$, and nothing changes. Now, we take this constraint and add in a term $t(x_0^2 - 1)$, and we add t under the max.

Now we minimize an unconstrained quadratic (homogeneous). We have a hidden constraint. If

$$f = \begin{pmatrix} x_0 \\ x \end{pmatrix}$$

then we can rewrite the inner min above:

$$\min \frac{1}{2} y^T \nabla^2 L(y, \lambda) y + \text{const}$$

and $\nabla^2 L(y, \lambda) \succeq 0$ with this constraint, the min is zero. So

$$d^* = \max_{\lambda \geq 0, \nabla^2 L(y, \lambda) \geq 0} \sum_{i=1}^{m} \lambda_i \alpha_i t$$

Now we just have to check what the Hessian will be:

$$\nabla^2 L \equiv \begin{bmatrix} 2t & \sum \lambda_i b_i^T \\ \sum \lambda_i b_i & Q_0 + \sum \lambda_i Q_i \end{bmatrix}$$

Equivalently, we have

$$\begin{bmatrix} -2t & -\sum \lambda_i b_i^T \\ -\sum \lambda_i b_i & -\sum \lambda_i Q_i \end{bmatrix} \preceq \begin{bmatrix} 0 & b_0^T \\ b_0 & Q_0 \end{bmatrix}$$

This is a special case of SDP:

$$\begin{array}{rl} \max & b^T w \\ s.t. & A^* w \preceq c \in \mathbf{S} \end{array}$$

In our case,

$$c = \begin{bmatrix} 0 & 0 \\ 0 & Q_0 \end{bmatrix}$$

Note that $A^* : \mathbb{R}^{m+1} \to \mathbf{S}^n$ is defined as being $w \mapsto w_0 A_0 + w_1 A_1 + w_2 A_2 + \cdots + w_{m+1} A_{m+1}$, so it's a linear combination of symmetric matrices A_i. This is exactly what we have above. For example, our first matrix would look like

$$A_0 = \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

and

$$A_1 = \begin{bmatrix} 0 & -b_1^T \\ -b_1 & -Q_1 \end{bmatrix}$$

What is the dual of this SDP? We can play the game theory problem again. We have the W player. The Lagrangian is

$$L(w, X) = b^T w + \langle X, c - A^* w \rangle$$

The dual problem, therefore, (the X player) has the minimization problem

$$\min_{X \succeq 0} \max_{w} L(w, X)$$

The trick is to move the adjoint around, as in linear:

$$\min_{X \succeq 0} \max_{w} \langle c, X \rangle + \langle w, b - AX \rangle \tag{12}$$

We want to find the map $A : \mathbf{S}^{n+1} \to \mathbb{R}^{m+1}$ that satisfies

$$\langle X, A^* w \rangle = \langle AX, w \rangle$$

If you run through the details, it turns out to be

$$A(X) = \begin{bmatrix} \langle A_0, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}$$

Because w is free in (12), we have the hidden constraint $b - AX = 0$ that we can add under the min:

$$\min_{X \succeq 0, b - AX = 0} \max_{w} \langle c, X \rangle + \langle w, b - AX \rangle \tag{13}$$

Then, we get for the dual player:

$$\begin{aligned} \min \quad & tr(CX) \\ s.t. \quad & AX = b \\ & X \succeq 0 \end{aligned}$$

And this looks just like the "star" for our LP.

For example, what is A_0X?

$$\langle C, X\rangle = trace\left(\left[\begin{array}{cc} 0 & b_0^T \\ b_0 & Q_0 \end{array}\right]\right)X \tag{14}$$

Typically in this kind of problem, we're doing a *lifting* from $\mathbb{R}^{n+1}$ to $\mathbf{S}^{n+1}$:

$$X = \left(\begin{array}{c} x_0 \\ x \end{array}\right)\left(\begin{array}{c} x_0 \\ x \end{array}\right)^T$$

This multiplication is rank 1, and this holds iff $X \succeq 0$ is rank 1. So then

$$\begin{aligned} tr(CX) &= trC\left(\begin{array}{c} x_0 \\ x \end{array}\right)\left(\begin{array}{c} x_0 \\ x \end{array}\right)^T \\ &= \left(\begin{array}{c} x_0^T \\ x \end{array}\right)C\left(\begin{array}{c} x_0 \\ x \end{array}\right) \\ &= q_0(X) \text{ if } x_0^2 = 1 \end{aligned}$$

Then (14) is equal to

$$x^TQ_0x + 2b_0^Txx_0$$

We have this $\langle A_0, X\rangle = 0$, and $AX \leq b$

$$\left[\begin{array}{c} \langle A_1, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{array}\right] \leq []$$

And $\langle A_0, X\rangle = -2x_{00}$ where

$$X = \left[\begin{array}{cccc} x_{00} & x_{10} & \cdots & x_{m0} \\ x_{10} & x_{11} & \cdots & x_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m0} & x_{m1} & \cdots & x_{m0} \end{array}\right] = \left[\begin{array}{c} x_0 \\ x \end{array}\right]\left[\begin{array}{c} x_0 \\ x \end{array}\right]^T = \left[\begin{array}{cc} x_0^2 & x_0x^T \\ x_0x & xx^T \end{array}\right]$$

We seem to have an error with the freeness of w, and there are probably some errors in the above for things that we left uncorrected on the chalkboard for this section of the notes.

So what happens to the trust region subproblem? We can think about the problem

$$\begin{array}{rl} min & q_0(x) \\ s.t. & q_1(x) \leq 0 \end{array}$$

Both of these functions could be non-convex. Look at the SDP relaxation. The surprising thing is that the gap is tight. Many ways to prove this. One way is to use Pataki-?. It's a condition on rank. You just count the number of constraints and variables you have. If you have a solution with rank 1, you can go backwards.

See if you can think of a way to reduce the rank. If you don't have a unique solution, there are some obvious things that you can do.

11 Duality and the Rayleigh Principle

Suppose we have a symmetric matrix A and we want to minimize

$$\begin{array}{rc} min & x^T Ax \\ s.t. & x^T x = 1 \end{array}$$

These points on a circle S^1 are definitiely not a convex set. We know that the anwer to this is $\lambda_{min}(A)$.

We do the usual:

$$\lambda_{min}(A) \geq \min_x L(x,y) = x^T Ax + \lambda(1 - x^T x)$$

If we look at the dual,

$$\lambda_{min}(A) \geq d^* = \max_\lambda \min_x L(x, \lambda)$$

$$\lambda_{min}(A) \geq d^* = \max_{\lambda, A - \lambda I \succeq 0} \min_x L(x, \lambda)$$

Then, we get the min at $x = 0$. So we get that the above is equal to

$$\max \lambda \text{ s.t. } A \succeq \lambda I = \lambda_{min}(A)$$

So $d^* = p^*$, strong duality.

11.1 The Trust Region Subproblem

Let's look at a more general problem. We'll examine

$$\begin{array}{rc} min & q_0(x) = x^T Ax + 2x^T x \\ s.t. & \|x\|^2 \leq \delta^2 \end{array}$$

If $A \succeq 0$ and we have a convex program, then Slater's condition holds (e.g. $\hat{x} = 0$), so we have strong duality. Without loss of generality, assume $\lambda_{min}(A) < 0$. Then

$$p^* = \min_{\|x\|^2 \leq \delta^2} x^T Ax + 2b^T x$$

The minimum is going to be on the boundary, so we can rewrite this with an equal:

$$p^* = \min_{\|x\|^2 = \delta^2} x^T Ax + 2b^T x$$

We can also write

$$p^* = \min_{\|x\|^2 \leq \delta^2} x^T (A - \gamma I)x + 2b^T x + \gamma x^T x$$

where $\gamma = \lambda_{min}(A) < 0$. By doing this, we have that this is PSD. Replacing with the equality,

$$p^* = \min_{\|x\|^2 = \delta^2} x^T (A - \gamma I)x + 2b^T x + \gamma \|x\|^2$$

since $\gamma < 0$.

Recall when you have an unconstrained problem, $\overline{x} \in argmin(f(x))$ implies $\nabla f(\overline{x}) = 0$ and $\nabla^2 f(\overline{x}) \succeq 0$. The Hessian is constant for a quadratic, so we'll find our minimum on the boundary. Our earlier expression is equal to

$$p^* = \min_{\|x\|^2=\delta^2} x^T(A-\gamma I)x + 2b^Tx + \gamma\delta^2$$

Now, replace back the other way

$$p^* = \min_{\|x\|^2\leq\delta^2} x^T(A-\gamma I)x + 2b^Tx + \gamma\delta^2$$

Because $A-\gamma I$ is singular, we can move towards the boundary without changing the quadratic. I can go in both directions (of the null space) and just move in the negative direction of the linear term b^Tx. $A-\gamma I \succeq 0$. So we've changed a non-convex problem to a convex problem. Because Slater's condition holds, we have strong duality. So, let's write down what the strong dual is:

$$\begin{aligned}
&\max_{\lambda\geq 0}\min_x x^T(A-\gamma I)x + 2b^Tx + \gamma\delta^2 + \lambda(x^Tx-\delta^2) \\
&= \max_{\lambda\geq 0}\min_x x^TAx + 2b^Tx + (\lambda-\gamma)(x^Tx-\delta^2) \\
&\leq \max_{\lambda\geq\gamma}\min_x x^T(A-\gamma I)x + 2b^Tx + \gamma\delta^2 + \lambda(x^Tx-\delta^2),\ \gamma<0 \\
&= d^* \leq p^*
\end{aligned}$$

So $p^* \leq p^*$ thus all inequalities here are tight: we have a zero duality gap. Does this prove attainment for d^*? I think we should get attainment here as well. I'll leave that as open, since I didn't think of that in advance.

So this problem can be solved by SDP, though there are more efficient ways to solve this that use sparse eigenvalue techniques. A lot of robust optimization works on this idea.

12 Robust Optimization

Robust optimization is a planning for the worst case. You can write such as a min-max problem. But if you can write the max as a min in the dual, then you just have a min problem, and that can be solved easily. Let's look at some examples.[6]

Our problem is minimize $\|Ax-b\|$ with uncertain A. One approach is to assume A is random at minimize the expectation of $\|Ax-b\|$. Another is by *worst-case*, minimize

$$\sup_{A\in\mathcal{A}} \|Ax-b\|$$

If you could write this problem as a strong duality using minimization, then you'd just have a min-min. This has been the hottest area in the last three

[6] This comes from Lecture II slides from the webpage. There are also citations to the book's pages.

years. One application is to robust LP, and trust topology in building bridges. It really started a lot of excitement in this area. Bridges are a lot more robust, and the big thing is that you can solve these problems.

As an example, take your uncertainty set to be $A(u) = A_0 + uA_1$. First, we minimize $\|A_0x - b\|_2^2$. Use the minimizer x_{norm}. x is fixed. As u changes,

$$r(u) = \|A(u)x_{norm} - b\|_2$$

changes. We can do the same with an x_{stoch}. Finally, we have x_{wc} for the worst-case approach.

In the stochastic approach, we have $A = \overline{A} + U$, and we want to minimize

$$\mathbb{E}\|(\overline{A} + U)x - b\|_2^2$$

In the worst case least squares, we take p matrices and we say that A varies by considering

$$\mathcal{A} = \{\overline{A} + u_1A_1 + \cdots u_pA_p \mid \|u\|_2 \leq 1\}$$

Then, we want to minimize

$$\sup_{A\in\mathcal{A}} \|Ax - b\|_2^2 = \sup_{\|u\|_2\leq 1} \|P(x)u + q(x)\|_2^2$$

In this problem, robust LP is equivalent to SDP.

A long time ago, before you were born, people use to heat rooms with radiators. Then came along the *bang-bang principle.*

Recall the trust region subproblem:

$$\begin{aligned} p^* = min \quad & x^TAx + 2x^Tx \\ s.t. \quad & x^Tx \leq \delta^2 \end{aligned}$$

We can add a new variable x_0

$$\begin{aligned} p^* = min \quad & x^TAx + 2x^Txx_0 \\ s.t. \quad & x^Tx \leq \delta^2 \\ & x_0^2 = 1 \end{aligned}$$

and we haven't changed anything, but everything is quadtratic. So, if we have a

$$y = \begin{bmatrix} x_0 \\ x \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

then a solution y for the second problem projects to a solution x for the first problem.

Let's look at the objective.

$$\min \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & b^T \\ b & A \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}$$

And we look at the constraint:

$$\begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} \leq \delta^2, x_0^2 = 1$$

So, we've just rewritten it. Everything is the same. Let's look at the Lagrangian. We'll examine the Lagrange dual d_{SDP} to be:

$$\max_{\lambda \geq 0; t} \min_{x_0, x} \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & b^T \\ b & A \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} + \lambda \left(\begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} - \delta^2 \right) + t(x_0^2 - 1)$$

$$= \max_{\lambda \geq 0; t} \min_{x_0; x} \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} t & b^T \\ b & A + \lambda I \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} - \delta^2 \lambda - t$$

So, what we have here is an SDP: Maximize $-\delta^2 \lambda - t$ subject to

$$\begin{bmatrix} t & b^T \\ b & A + \lambda I \end{bmatrix} \succeq 0, \lambda \geq 0 \tag{15}$$

Now, let's look at p^*_{SDP}, the primal SDP. First, let me just rewrite the constraint (15) a little bit:

$$t \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} + \lambda \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \preceq \begin{bmatrix} 0 & b^T \\ b & A \end{bmatrix}$$

So,

$$p^*_{SDP} = \min \left\langle \begin{bmatrix} 0 & b^T \\ b & A \end{bmatrix}, X \right\rangle$$

such that

$$\left\langle \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, X \right\rangle = -1$$

$$\left\langle \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}, X \right\rangle \geq -\delta^2$$

$$X \succeq 0$$

So we have strong duality

$$p^* \geq d^*_{SDP} = p^*_{SDP}$$

For more general programming, we have **strict** inequality! So, if X is of the form

$$X = \begin{bmatrix} x_0 \\ x \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}^T$$

(that is, $\operatorname{rank}(X) = 1$), then we have the original problem.

13 Geometric Problems

I'd like to go through these quickly. Book pages and slides are available. There are several additional problems in the book pages.

13.1 Closest point to a convex set

One of the interesting things about this projection problem: This is an interesting way of interpreting duality. In the primal problem, the optimum is the closest point $x \in C$ from a point $x_0 \notin C$ outside the set C. The dual problem is interesting algebraically and geometrically:

If you take any hyperplane H separating C from x_0, and say we call d_H the distance from the point x_0 to H, then the dual problem is to maximmize d_H.

The optimal distance d_H^* corresponds to the nearest point $x^* \in C$.

13.2 Separating a point from a convex set

All you have to do is find the closest point (see Section 13.1).

13.3 Distance between sets

This is a generalization of the problem in Section 13.1 (from a set and a point to a set and a set).

13.4 Minimum volume ellipsoids around a set

The *Löwner-John ellipsoid* of a set C is the minimum value ellipsoid $\mathcal{E}$ such that $C \subseteq \mathcal{E}$. We parameterize $\mathcal{E}$ as $\mathcal{E} = \{v \mid \|Av + b\|_2 \leq 1\}$. Without loss of generality, we assume that $A \in \mathbf{S}_{++}^n$. The volume of $\mathcal{E}$ is proportional to $\det A^{-1}$. To compute the minimum volume ellipsoid: Minimize $\log\det A^{-1}$ subject to

$$\sup_{v \in C} \|Av + b\|_2 \leq 1$$

For a finite set $C = \{x_1, \ldots, x_n\}$, this is computable.

13.5 Maximum volume inscribed ellipsoid

We parameterize $\mathcal{E}$ as $\mathcal{E} = \{Bu + d \ \mid \|u\|_2 \leq 1\}$. Again, we can assume that $B \in \mathbf{S}_{++}^n$. We can compute by solving:

$$\begin{array}{rc} max & \log\det B \\ s.t. & \sup_{\|u\|_2 \leq 1} I_C(Bu + d) \leq 0 \end{array}$$

If $f(B) = \log\det B, B \succ 0$, then

$$\frac{\partial f(B)}{\partial B} = \left(\frac{1}{\det B}\right)\left(\frac{\partial}{\partial B}\det B\right) = B^{-1},$$

with the last equality coming from Cramer's rule. That is the matrix of cofactors (other books call it an adjoint). So, $\frac{\partial}{\partial B_{ij}}$ is the ij-cofactor (expand along the row).

How do we find

$$\frac{\partial B^{-1}}{\partial B}?$$

The trick is to use $BB^{-1} = I$ and use the product rule.

13.6 Centering

One centering approach is to find the "Chebyshev center." Another approach is the center of ellipsoids. We can also look for an analytic center.

13.7 Linear Discrimination

Suppose you want to separate tow sets of points $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_M\}$ by a hyperplane H so that the points on X lie on one side of H and the points of Y lie on the opposite side.

13.7.1 Robust Linear Discrimination

Instead of just finding some separator, you want to find the one in the middle that separates as best as possible, so we want to maximize the norm of the separator. This problem is a QP.

13.7.2 Approximate Linear Separation of Non-Separable Sets

Sometimes you can't separate, so we want to minimize the error. This problem allows us to do an approximate linear separation. This is a heuristic.

13.7.3 Support vector classifier

13.8 Nonlinear Discrimination

Linear models are the easiest to use, but nowadays, we can try to use nonlinear separators. There are a family $F = (F_1, \ldots, F_k) : \mathbb{R}^n \to \mathbb{R}^k$ that serve as the basis functions.

13.9 Placement and Facility Location

We mentioned the flow problem. This is simpler than the quadratic assignment problem because of a specialize flow cost.[7]

[7] A group is going to present this afternoon. Please come and support "the machine." Are people interested in making T-shirts?

14 Unconstrained Minimization

If you were taking a detailed class in constrained optimization, typically they start with linear programming, then do the non-linear case (starting with unconstrained problems).

We have the problem

$$\text{Minimize } f(x)$$

The best problems are when f is convex and in C^2. For these types of problems, we assume that the optimal value

$$p^* = \inf_x f(x)$$

is attained and is finite. The methods:

- Produce a sequence of points $\in$ *dom*(f)
- Iterate

We require a starting point $x^{(0)}$, and the sublevel set of $x^{(0)}$ is closed.

Something that helps is the notion of *strong convexity*, namely,

$$\text{There exists an } m > 0 \text{ such that } \nabla^2 f(x) \succeq mI \text{ for all } x \in S$$

A classic book on unconstrained optimization is by Dennin-Schnabel. This text includes a very careful analysis on stopping criteria. These are subtle things: you want your stopping criteria to be scale-free.

14.1 Descent Methods

Many of the algorithms in the literature are called *descent methods*. You have your current point $x^{(k)}$. Somehow, you find a search direction $\Delta x^{(k)}$, then you find your step length $t^{(k)}$. Now modern (1960's) optimization says that you want to have a good search direction (and spend your time working hard in finding the search direction), then, the step length needs to be relatively cheap. This because step-length involves a lot of function evaluations. Most of your time in interior methods is in finding this search direction. (Then you change μ, meaning your trying to improve on that search direction.)

The other things mentioned on this slide is the *descent direction*: you want to ensure your objective is decreasing, that is,

$$\nabla f(x)^T \Delta x < 0.$$

If $g(t) = f(x + t\Delta x)$ then $g'(t) = \nabla f(x + t\Delta x)^T \Delta x$ and in particular, $g'(0) = \nabla f(x)^T \Delta x < 0$.

We want to guarantee that we get sufficient decrease in an iteration. we want to go to a point that is sufficiently less steep.

A new development is something that's called *filter methods*. These do not guarantee decrease, so this is a strange phenomenon. They mostly come up in constrained optimization, so I won't discuss it now: we'll talk about them more later. There are certain methods that work well that allow an increase once in a while. This has been an important idea in the last couple of years, and it has changed some of how we look at unconstrained optimization.

14.1.1 Gradient Descent Method

This is also called Cauchy's descent method. We simply go in the search direction $\Delta x := -\nabla f(x)$. If you want to find this steepest descent, we're really solving this minimization

$$min \quad d^T \nabla f(x^k) \quad (\ + \ f(x^k) \ )$$

This is a linear model. The optimal d here is? We need some kind of bound on this, because it's unbounded. How about we amend this to:

$$\min_{\frac{1}{2}\|d\|^2 \leq 1} \quad d^T \nabla f(x^k) \quad (\ + \ f(x^k) \ )$$

$d = 1$ is a Slater point, so Slater's condition holds. The Lagrangian for this problem is

$$L(d, \lambda) := d^T \nabla f(x^k) + \frac{\lambda}{2}(\|d\|^2 - 1) \tag{16}$$

$$0 = \nabla L \equiv \nabla f(x^k) + \lambda d$$

So $\lambda \geq 0$, so $\Longrightarrow d = \frac{-1}{\lambda} \nabla f(x^k)$. Thus, $\lambda = \|\nabla f(x^k)\|$.

MatLAB has a very nice demonstration of this, called the *banana function*:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

After turning off MOSEK, we can use steepest descent. Steepest descent goes across the valley.

The basic idea for the simplex method (it's non-differentiable: it just uses function values) is:

- Evaluate f at each vertex of simplex
- Find largest value v_0
- Reflect on the link of v_0.
- Check the new point and act like a trust-region problem (if it does not do well, then shrink.)

So it's basically trying to improve the worst value of the simplex. So as you move along, you have the property that the worst value of the simplex gets better and better. This is becoming hot because it may be too expensive (or impossible) to do gradients. Look for papers by Nelder-Mead. IBM has some people who are working on this, and they have some production code.

There's some very nice results on the convergence for steepest descent. The number of "wiggles" are based on something called the Kantorovich lemma.

Now, if you were to change the norm in (16), you would change the search direction. So, scaling is very important for the steepest descent method. What's the best norm? Which is the best p to choose? Note, you can also change which p you use in the course of the algorithm.

14.2 Newton's Method

Newton's method is based on a quadratic model:

$$\min \quad d^T \nabla f(x^k) + \tfrac{1}{2} d^T \nabla f(x^k) d \quad (+ f(x^k) \ \ )$$

We set the derivative equal to zero for a *stationary point*, we get

$$\nabla^2 f(x^k) d + \nabla f(x^k) = 0$$

Note that the d here is jut ΔX_{nt} in the slides. It's the best quadratic polynomial $\hat{f}$ fitting f at x. Finding a min of this quadratic polynomial $\hat{f}$ is as simple as finding a zero of $\hat{f}'$, a linear function.

If $\nabla^2 f(x) \succeq 0$, then

$$\begin{aligned} g'(0) &= \nabla f(x)^T \Delta x_{nt} \\ &= -\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) < 0 \end{aligned}$$

What is the best p? The best p is just the Hessian. The thing to think about is, "What does a scaling mean? What is the best scaling?" This gives you Newton's method. Moreover, it shows that it's scale invariant (since you can't do any better).

14.3 Filter methods in constrained optimization

Suppose you have a non-linear least squares problem: you have some sort of non-linear functions that you want to fit to the data. Then you are looking

$$\min \quad \textstyle\sum_{i=1}^n (r_i(x))^2 \quad , \quad m \text{ objectives}$$

Another thing you might consider is putting weights w_i on the sum

$$\min \quad \textstyle\sum_{i=1}^n w_i (r_i(x))^2 \quad , \quad m \text{ objectives}$$

Some of you may have heard of multi-criteria optimization. You may want to treat this as a multi-criteria problem. You want to find points which are not dominated by other points.

14.3.1 A Filter Algorithm

Here's how we could look at solving this. We ask in

$$x^{k+1} = x^k + \Delta x^k$$

when is x^{k+1} is better than x^k? The vector $y^{k+1} = \big(r_i(x^{k+1})^2\big)$ is better than (or *dominates*) the vector $y^k = \big(r_i(x^k)^2\big)$ if $r_i(x^{k+1})^2 < r_i(x^k)^2$ for all i.

The basic idea is to build a numerical method where non-dominated points are accepted.

14.4 Condition Number

The condition number of a positive definite matrix Q is

$$\frac{\lambda_{max}(Q)}{\lambda_{min}(Q)} \geq 1$$

The identity I is the best that you can do. If you're on a circle, steepest descent means that you're going to converge in one iteration. How do you take a quadratic that looks like

$$\frac{1}{2}x^T Q x + x^T x$$

You scale the problem with $x \leftarrow Sy + d$. What happens when you make this substitution for x? You're going to get a quadratic

$$\frac{1}{2}y^T S^T Q S y + \cdots$$

What's the best S that you can choose? We want the Hessian to be the identity, so the best scaling is $S = Q^{-\frac{1}{2}}$.

14.5 Steepest Descent and Deflected Gradient Methods

$\Delta x^k = -B\nabla f(x^k)$ is called a *deflected gradient.*

$$\begin{aligned}(\Delta x^k)^T \nabla f(x^k) &= -\nabla f(x^k)^T B \nabla f(x^k)\\ &< 0 \text{ if } B \succ 0\end{aligned}$$

One way of deriving Newton's method is to say, "I want the best local scaling." Newton's method gives you the best local scaling. In particular, if you scale it again, nothing's going to change. Thus, it's *scale free*: Many optimization problems (like linear programming) say you're working with units of cars/airplanes, and the price is on the order of millions, the change of scale causes magnitude problems. If you do steepest descent, you're going to zig-zag forever. But in Newton's method, you're going to be essentially doing the same thing. You choose B to be the inverse of the Hessian.

Steepest descent gives us linear convergence: $\|x^{k+1} - x^k\| \leq \gamma \|x^k - x^{k-1}\|$ for $0 \leq \gamma < 1$. Newton's method gives us quadratic convergence. Quadratic convergence means $\|x^{k+1} - x^k\| \leq \gamma^2 \|x^k - x^{k-1}\|$ for $0 \leq \gamma < 1$. Convergence analysis is a very interesting area.

When I was a graduate student, we didn't have the web. I was looking for a paper in Math Programming in the library. I found a paper there published by "Anonymous". It was submitted by Phillip Wolfe. The problem is finding a minimum without any assumptions. One of the algorithms it goes through is to use a countable dense subset. There's a difference between proving convergence and proving convergence for an algorithm on a computer.

Steepest descent in the local Hessian norm is one way of deriving Newton's method. Locally, you have just circles. We have a strong lower bound on the smallest eigenvalue.

Algorithm (Newton's Method)

Given a starting point $x \in dom(f)$ and tolerance $\epsilon > 0$,

1. Compute the Newton step and decrement:

$$\Delta_{nt} := -\nabla^2 f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$$

2. Stopping criterion: **quit** if $\lambda^2/2 \leq \epsilon$
3. Line search: Choose step size t by backtracking line search
4. Update: $x := x + t\Delta x_{nt}$.

Inexact Newton's Method means to solve $B_k \Delta x^k = -\nabla f(x_k)$ approximately. You have to decide what approximately means.

Super linear convergence means $\|x^{k+1} - x^k\| \leq \gamma_k \|x^k - x^{k-1}\|$ for $0 \leq \gamma_k < 1$, and $\gamma_k \to 0$. You want to get at least this, as quadratic convergence may be too much to ask for. These "quasi-Newton methods" use finite differences.

Newton's method is affine invariant (that is, it is independent of linear changes of coordinates.)

14.6 Classical Convergence Analysis

This is due to Kantorovich[8]. With some error estimates, you can prove quadratic convergence:

$$\frac{L}{2m^2} \|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{L}{2m^2} \|\nabla f(x^{(k)})\|_2 \right)$$

[8] Khachian (Kantorovich's student) visited Waterloo (while he was still alive). Kantorovich said that we was approached by the KGB with a transportation problem. They approached him a second year. They wanted something better than optimal. There is a moral here: How do you satisfy the manager? You don't give them the optimal solution.

There's a *damped Newton method.* If you do this, you'll lose quadratic convergence.

In conclusion, you get (locally) quadratic convergence in Newton's method, and quadratic convergence is very fast.

Far from the solution you want a combination of Newton's Method and Steepest Descent. So, this is the trust region problem. Newton's Method doesn't know the difference between a min and a max.

There are some strong convergence results for strictly convex functions and you need the smallest eigenvalue of the Hessian. (If the Hessian isn't invertible at your point, you do a steepest descent.)

The exact line search has a step length of 1, asymptotically. You lose quadratic convergence unless you have a step length of 1 asymptotically.

This started a few years ago. Many of us have joined the NLF (Newton Liberation Front). Don't hold Newton back. Because we have a strongly convex function, we have a nice positive definite Hessian at the optimum, and we have a nice Taylor series at that point. So, this provides us the second-order sufficient condition for optimality.

Even though Newton's method is scale-free, a local region is not scale-free. So, there is a problem: The difficulty is you create quadratic convergence, but you don't know that region. So, what can you do about that? There is a break through that coincided with the interior point revolution:

14.7 Self-Concordance

This came about in the late 1980s from Nesterov and Nemirovski. Nesterov, one of the developers on this, is one of the people who worked on the ellipsoid method.

The idea is to deal with this problem in Newton's method. Can we enlarge this region of local quadratic convergence somehow? Each line in a Russian paper takes an hour to understand. This is one of those lines: A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is *self-concordant* if

$$|f'''(x)| \leq 2f''(x)^{3/2} \text{ for all } x \in dom(f)$$

This says that the second derivative can't change very much. It avoids the problem of needing this smallest eigenvalue. There are many examples of self-concordant functions:

- linear and quadratic functions
- negative logarithmic functions

For a good resource on this, pick up the text by James Renegar.

Interior point methods use barrier functions. If we just look at self-concordant barrier functions, the best one is the log-barrier one that we use. The book by Nesterov and Nemirovski give the answer why.

Usually, the theory is quite different from the practice. For example, our upper bounds on number of iterations is usually very far from being tight.

14.8 Implementing Newton's Method

In numerical analysis, you **never** invert. You solve a system of equations

$$H\Delta x = g.$$

You do a Cholesky factorization. There's so much on Cholesky factorization that you can solve surprising large problems. MatLAB can tell you if your matrix is PSD.

15 Equality Constrained Minimization

We started with unconstrained, then we move from unconstrained to equality constraints (and eventually to inequalities).

Consider f is a convex, twice differentiable function and

$$\begin{array}{rc} \min & f(x) \\ \mathit{sub.to} & Ax = b \end{array}$$

We assume that p^* is finite and attained, and that A is a full-rank matrix. Complementary slackness still holds. Our optimality conditions are: x^* is optimal iff there is a ν^* such that

$$\nabla f(x^*) + A^T\nu^* = 0, Ax^* = b$$

Because we have a convex problem, we can combine the two conditions:

$$\begin{array}{rc} \min & \frac{1}{2}x^TPx + q^Tx + r \\ \mathit{sub.to} & Ax = b \end{array}$$

with optimality condition

$$\left[\begin{array}{cc} P & A^T \\ A & 0 \end{array}\right]\left[\begin{array}{c} q^* \\ \nu^* \end{array}\right] = \left[\begin{array}{c} -q \\ b \end{array}\right]$$

The coefficient matrix is called the KKT matrix. It is non singular iff $Ax = 0, x \neq 0 \Longrightarrow x^TPx > 0$.

Another way of approaching this problem is to use *null space techniques*. You find a basis for the null space F and rewrite the solution set of $Ax = b$ as

$$\{x|Ax = b\} = \{\hat{x} + Fz : z \in \mathbb{R}^{n-p}\}$$

Here, $\hat{x}$ is a particular solution. So, can you do this efficiently?

Sometimes it's very useful to work with the dual simplex method for LP. What happens very often is that you have an LP and you solve it and you get an optimal solution. Sometimes, as mentioned, your data changes a bit. Your resources change (that is, your right hand side b changes). So you have dual feasibility, and you've lost primal feasibility. You still have complementary slackness. Why not work with what you have and recover? Now, we minimize $f(Fz + \hat{x})$. From the solution z^*, we obtain x^* and ν^* as

$$x^* = Fx^* + \hat{x} \qquad v^* = -(AA^T)^{-1}A\nabla f(x^*)$$

15.0.1 Example: Separable problems

A problem is separable if it is of the form

$$\begin{array}{rc} \min & f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \\ s.t. & x_1 + x_2 + \cdots + x_n = b \end{array}$$

We have the Newton decrement[9]

$$\lambda(x) = \left(\Delta x_m^T \nabla^2 f(x) \Delta x_n\right)^{1/2}$$

ALGORITHM (Newton's Method with Equality Constraints)
Given a starting point $x \in dom(f)$ with $Ax = b$ and tolerance $\epsilon > 0$,

1. Compute the Newton step and decrement: $\Delta x_{nt}, \lambda(x)$
2. Stopping criterion: **quit** if $\lambda^2/2 \leq \epsilon$
3. Line search: Choose step size t by backtracking line search
4. Update: $x := x + t\Delta x_{nt}$.

15.1 Solving KKT Systems

Start with a system

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = - \begin{bmatrix} g \\ h \end{bmatrix}$$

We can solve this by LDL^T factorization, elimination (if H is nonsingular), or a special elimination if H is singular.

The complexity per iteration of each of the three methods is identical. You might choose one method over the other, say, based on the number of primal versus dual variables.

15.1.1 Example: Network Flow Optimization

(Why do we always throw away the last row in these matrices?)

15.1.2 Example: Ananlytic Center of Linear Matrix Inequality

16 Nonlinearly Constrained Optimization

The basic tool that we're going to use is Lagrange multipliers:

$$L(x, \lambda) = f(x) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i c_i(x).$$

[9] sounds like an insult

We have the notation here that

$$\mathcal{A}^* = \{i \in \mathcal{I} : c_i(x^*) = 0\} \cup \mathcal{E}$$

Complementary slackness says that $\lambda_i^* c_i(x^*) = 0 \ \forall \ i \in \mathcal{E}$, which says that either the inequality is zero, or the constraint is zero. That is, either $\lambda_i^* = 0$ or c_i is active at x^*. For the interior point methods, we add these complementary slackness, because it gives us the right number of equations.

For these methods, it's better work with the active set. We require the *strong constraint qualification*, which asks for linear independence. It says that this set of multipliers is going to be unique at the optimum. This is **stronger** than the Magnasarian-Fromovitz conditions. In this weaker setting, we don't have that the Lagrange multipliers are unique. If the mulitpliers are unique, you have (in some sense) a robust problem.

If you have a system of equations

$$Ax = b$$

then the classical definition of an ill-posed problem is when this solution x is not unique. It is a well-posed problem when this solution is unique. For example, all sorts of problems can arise when you're working with a singular matrix A. What do you do, you typically add a constraint like $\|x\| \leq \delta$. But now, your set of solution lies in a compact set. This is called *regularization*. Now that you have a compact set, you can say something better about the solution. This has come up in things like CAT scans. A CAT scan is an infinite-dimensional problem. It has a compact linear operator

$$\int x(t)\psi(t) = b$$

It's a Fredholm operator of the first kind working an a compact operator $\mathcal{K}(x) = b$. So we perturb the right side a bit

$$\int x(t)\psi(t) = b + \epsilon$$

Because the inverse is unbounded, this should blow up. The other problem is, the Lagrange multiplier may not exist.

Constraint qualifications are very important. You probably saw Lagrange multpliers and then quickly forgot about it. One thing that we didn't see before are the second order conditions. The sufficient second-order condition for constrained optimization is

$$w^T \nabla_{xx}^2 L(x^*, \lambda^*) w \succ 0 \tag{17}$$

It makes sense that if you're just looking at $\overline{x} \in argmin(f(x))$, then you have $\nabla f(\overline{x}) = 0$ and $\nabla^2 f(\overline{x}) \succeq 0$. And you can go back the other way: $\nabla f(\overline{x}) = 0$ and $\nabla^2 f(\overline{x}) \succ 0$ imply $\overline{x} \in argmin(f)$, and this is strict-local. So, when we were minimizing $f(x) = x^4$, the Hessian was not PSD at $x = 0$. Here, the necessary conditions hold, but the second-order sufficient conditions don't hold, so we lose quadratic convergence. Why? $\nabla^2 f(0) \not\succ 0$. Equation (17) must hold on the feasible set.

Strict complementarity says that $\lambda_i^* - c_i(x^*) > 0$ for all i. So, one of these is not going to be 0. If we don't have strict complementary slackness, you'll have some redundancy. We can look at an example from linear programming: This means that a vertex is not *simply determined.* You can have multiple optimal solutions and multiple bases for a vertex. It is a theorem of Hoffman and Goldman (or maybe Tucker?) that there always exists a solution that satisfies strict complementarity. This is what makes interior point methods work so well for linear programming. For semi-definite programming, this is not true. You can actually construct problems where strict complementarity fails, and you can see the problem when you try interior point methods on these: loss of numerical accuracy, etc. Why? It's because the Jacobian is singular there. Gale (a well-known economist) has said that the Lagrange multipliers are one of the important tools in economics[10].

16.1 Sequential Quadratic Programming

We're going to do something that's very much like Newton's method. At a current point x_k and a current estimate for the Lagrange multipliers, we can from the Lagrange multpliers formulate our next step strangely: we take the gradient of the objective and the Hessian of the Lagrangian instead of the gradient of the Lagrangian.

$$q_k(d) = \nabla f(x_k)^T + \frac{1}{2} d^T \nabla^2_{xx} L(x_k, \lambda_k) d$$

Why should we possibly want to do this? Then, we replace the constraint function by linear approximations. When you for the Lagrangian of this problem, you have Lagrangian multipliers on the values of the gradient.

The step d_k can be found by solving this quadratic:

$$\min\{c_i(x_k) + \nabla c_i(x_k)^T d \leq 0, i \in \mathcal{I} \qquad c_i(x_k) + \nabla c_i(x_k)^T d = 0, i \in \mathcal{E}\}$$

What are the optimality conditions for this quadratic? We have the Lagrangian:

$$L(d, \mu) = q(d) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \mu_i (c_i(x) + \nabla c_i(x)^T d)$$
$$\nabla L(d, \mu) = \nabla q(d) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \mu_i \nabla c_i(x) = 0$$

Let's just pretend that we have equality constraints, because it'll just make our lives easier. Then we don't have to deal with extra notation that will just cause

[10] Does anybody do economics here? No? Okay, then I can tell you whatever I want!

a headache. Pretend $\mathcal{I} = \emptyset$:

$$\begin{aligned} L(d,\mu) &= q(d) + \sum_{i\in\mathcal{E}} \mu_i(c_i(x) + \nabla c_i(x)^T d) \\ \nabla L(d,\mu) &= \nabla q(d) + \sum_{i\in\mathcal{E}} \mu_i \nabla c_i(x) = 0 \qquad \text{Dual Feasibility} \\ c_i(x) &+ \nabla c_i(x)^T d = 0,\ \forall\ i \qquad \text{Primal feasibility} \end{aligned}$$

So we have

$$\nabla f(x) + \nabla^2_{xxx} L(x,\lambda) d + \sum_{i\in\mathcal{E}} \mu_i \nabla c_i(x) = 0$$

$$\nabla f(x) + \left[\nabla^2 f(x) + \sum_i \lambda_i \nabla^2 c_i(x)\right] d + \sum_{i\in\mathcal{E}} \mu_i \nabla c_i(x) = 0$$

$$\nabla L(x,\lambda) = \nabla f(x) + \sum_{i\in\mathcal{E}} \lambda_i \nabla c_i(x) = 0 \qquad c_i(x) = 0 \forall i \tag{18}$$

thus,

$$F(x,\lambda) = 0. \tag{19}$$

So, we apply Newton, and we get (asymptotic) quadratic convergence, that is

$$F'(x,\lambda) d = F(x,\lambda)$$

If we diffferentiated again, we'd get the same thing. All we have to do is go redo this with the case that we have a mix of equality and inequality constraints.

So, we have

$$F'(x,\lambda) = \left[\begin{array}{cc} \nabla^2 d(x) + \sum_{i\in\mathcal{E}} \lambda_i \nabla^2 c_i(x) & \nabla c(x)^T \\ \nabla c(x) & 0 \end{array} \right]$$

where $c : \mathbb{R}^n \to \mathbb{R}^m$. The only difference is

$$\begin{aligned} d &\equiv \Delta x \\ \mu &\equiv \Delta\lambda, \qquad \mu = \lambda + \Delta\lambda \end{aligned}$$

So, we can take

$$F'(x,\lambda) \left[\begin{array}{c} \Delta x \\ \Delta\lambda \end{array} \right] = F(x,\lambda)$$

and we rewrite. We'll get exactly the same thing except...

In order to analyze Newton's with SQP, all we're doing is trying to solve the equation (19).

In solving the BFGS, there's what's called a *rank* 2 ... and in updating the matrices B_k by

$$B_{k+1} = B_k - \cdots \tag{20}$$

we maintain positive semi-definiteness.

B does not exactly approximate the Hessian $\nabla^2 f(x)$. So, it approximates the behavior of the Hessian (on the important points). But it doesn't converge to the Hessian. There's also a way to make sure that the update of B is not too hard. So the formula (20) is not too hard to implement. In practice, the BFGS is empirically better than the DFP.

What are the convergence properties of the basic SQP? Far away from the optimum, you have to do a line search. You can't just take steps of lengths one. A line search is what? You have to satisfy the constraints as well in finding a direction. The constraints can be very strange as well. You might just have "two whiskers." There's no way that you're going to stay feasible. Especially because of equality, we use what's called a *merit function.* In constrained optimization, you have the objective **and** some other functions.

One of the things that these algorithms suffer from is called the *Maratos effect.* What could happen is that your feasible set might be on some convex set, and you might start (at x_0) not on the optimal set. Suppose you have a lot of constraints, and they are penalized a lot compared to your objective, but you don't know the scaling. You may have this property that you get closer and closer to the feasible set, and eventually you get around to x^*, but what happens on many of these methods with *exterior penalty functions*, they only get feasible in the end. We get "close to feasible" too soon. It follows the feasible boundary, and it takes forever to get to the optimal solution.

There are many merit functions that you can use. A natural one you might think of is the *augmented Lagrangian.* One of them might be something like

$$f(x) + w\left[\sum_{i\in\mathcal{E}} c_i^2(x) + \sum_{i\in\mathcal{I}} \max{}^2(0, c_i(x))\right].$$

We're taking an ℓ_2-norm on the violation of the constraints. We say that w is a *penalty.* It's very hard to decide which to do first: do you want to get close to feasible first or head towards the true objective $f(x)$ first?

There is an ℓ_1 merit function

$$f(x) + w\left[\sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \max\,(0, c_i(x))\right].$$

You can prove that you only get feasible in the limit with these types of functions.

SQP beats interior point methods. For convex problems, the interior point methods become superior. Besides the Meritos effect, there's another difficulty: The original subproblem (the QP subproblem) may be bounded below. One of the ways of handling that is by the trust region subproblem:

$$\|d\| \le \delta_k.$$

This looks good: you get the best of both worlds: The Lagrangian will be $\succeq 0$. But, does anybody see a difficulty? You fix one thing, and something else comes out. The problem may become infeasbile (these $c(x_k)$'s are constant). You may have made your ball small enough that it doesn't intersect the feasible set to our auxilliary problem!! What do people do then? You have to be careful what you change, or you may lose a lot. One of the things that people do (again, let's just look at the equality case) is they replace those equality constraints by a new constraint

$$\sum \left(c_i(x_k) + \nabla c_i(x_k)^T d\right)^2 \leq \text{ something.}$$

In other words, they want to get a least squares solution. You have several quadratic constraints and a quadratic objective. This is called the *CDT problem.* So, now you have the two-trust region subproblem (2TRS). We're starting to get from a nice motivation and a nice picture to all sorts of complications. We took a quadratic approximation of the Lagrangian but a linear approximation of the constraints. We worry about unboundedness, infeasibility, and so on. So, people keep trying to fix all of these problems. There is one simple fix for all of these difficulties.

The simple fix is to consider the problem:

$$\begin{array}{rl} min & q_0(d) \approx f(x_k + d) \\ s.t. & q_i(d) \approx c_i(x_k + d) \end{array}$$

(QCQP) or (QQP). We have all quadratics. The q_i are all ≤ 0 or $= 0$. We add $\|d\|^2 \leq \delta_k^2$. Now, all our problems are fixed. No Meritos effect, no infeasibility. But, there's still a problem! This problem is too hard to solve: this is NP-hard in general. So, we use a Lagrangian relaxation, that is, an SDP!

So, that was a quick hour-and-a-half on non-linear programming. We'll finish off with the interior point method for the convex case, just to look at that again.

17 Interior Point Methods

We reformulate our problem via an indicator function

$$\begin{array}{rc} \min & f_0(x) + \sum_{i=1}^m I_-(f_i(x)) \\ s.t. & Ax = b \end{array}$$

We have our nice convex logarithmic barrier function

$$\phi(x) = -\sum_{i=1}^m \log(-f_i(x)).$$

ϕ is twice continuously-differentiable, and it has a very nice second derivative.

Fiacco[11] and McCormick developed the SUMT. If you have inequality con-

[11] When I was a grad student, we were given a very hard problem to solve. The memory available was about 100K. If you begged, then you were able to use `whatbig`, which used 150K of memory. I couldn't solve this problem so I sent Fiacco a letter, and he sent me the code as a package in the mail (consisting of 500 computer cards). His code worked. It was like a sledgehammer. His code was actually polynomial time on linear programming problems.

straints, you use this log-barrier to keep you feasible. Why did this lose popularity so quickly?

In the second derivative, we have rank one matrix, so $\nabla^2\phi$ is going to be singular. You're trying to solve a system of linear equations where the matrix is **very** ill-conditioned. Fiacco and McCormick did something else in their code which avoided this problem. But, we've cycled back! Now, people are using this to solve convex and non-convex problems.

17.1 Central Path

What do we do? For $t > 0$, define $x^*(t)$ as the solution of

$$\begin{array}{cc} min & tf_0(x) + \phi(x) \\ s.t. & Ax = b \end{array}$$

Then, the central path is $\{x^*(t) : t > 0\}$. We can also look at dual points on the central path. We can look at a weak-duality type of argument

$$p^* \geq g(\lambda^*(t), \nu^*(t)).$$

What happens is that some of the terms go away. We're assuming that we have feasibility $Ax = b$. So, we have this $\frac{1}{t}$ because of how they defined the λ_i.

17.2 Interpretation using the KKT conditions

We differentiate the log-barrier problem, either the primal or the dual. So condition 3 (in slides) replaces the complementary slackness $\lambda_i f_i(x) = 0$ with

$$-\lambda_i f_(x) = \frac{1}{t}.$$

17.3 Force field interpretation

We have the centering problem, and the force field interpretation is that the first term is a *potential of force field* F_1.

17.3.1 Example: Linear programming

There are forces that are forcing the current point x that are pulling it. Some are pulling it towards optimality, and some are pulling towards the interior regions. These things simply all cancel where x^* is. So the sum of all of those forces $F_0(x) = -tc$ and the constraint forces

$$F_i(x) = \frac{-a_i}{b_i - a_i^T x}$$

should all cancel out at optimality.

Here, we have the barrier method:

Algorithm (Barrier Method)

Given strictly feasibly x, $t := t^{(0)} > 0$, $\mu > 1$ and a tolerance $\epsilon > 0$, repeat:

1. *Centering step.* Compute $x^*(t)$ by minimizing $tf_0 + \phi$ subject to $Ax = b$.
2. *Update.* $x := x^*(t)$.
3. *Stopping criterion.* **QUIT** if $m/t < \epsilon$.
4. *Increase t.* $t := \mu t$.

If I pick $\sigma = 1$, I'm heading towards the central path (with zero duality gap). If I pick $\sigma = 0$, I'm heading towards the optimum (with zero duality gap). But another thing you can do is try the *predictor-corrector method*, where you keep alternating your σ between 1 and 0. There is a way to matrix factor to do both steps at once. You can solve DE's using predictor-corrector[12].

The number of iterations for interior point methods is (largely) independent of dimension. This is very different if you use dual methods: the cost of iteration is much less, but the number of iterations grows.

17.4 Generalized Inequalities

We assume that we have a family $f_i(x) \preceq_K 0$ of K-convex functions.

17.5 Log Barrier and Central Path

17.5.1 Dual points on Central Path

17.6 Example: Semi-Definite Programming

In the constraints

$$F(x) = \sum_{i=1}^{n} x_i F_i + G \preceq 0$$

the portion $\sum_{i=1}^{n} x_i F_i$ is our $A^*(x)$. So ,w ehave $C = -G$, and we can rewrite it in the form

$$A^*(x) \preceq C.$$

So, I would look at $\log\det(C - A^*(x))$. Differentiating is a little more complicated.

Our central path: $x^*(t)$ minimize $tc^T x - \log\det(-F(x))$, hence

$$tc_i - tr(F - iF(x^*(t))^{-1}) = 0, \qquad i = 1, \ldots, n$$

The duality gap on the central path:

$$c^T x^*(t) - tr(GZ^*(t)) = p/t.$$

[12] I'm tiring out. I usually tire out when people aren't listening to me.

ALGORITHM (Barrier Method for SDP)

Given strictly feasibly x, $t := t^{(0)} > 0$, $\mu > 1$ and a tolerance $\epsilon > 0$, repeat:

1. *Centering step.* Compute $x^*(t)$ by minimizing $tf_0 + \phi$ subject to $Ax = b$.
2. *Update.* $x := x^*(t)$.
3. *Stopping criterion.* **QUIT** if $\left(\sum_i \theta_i\right)/t < \epsilon$.
4. *Increase t.* $t := \mu t$.

The only difference is the stopping criteria.

18 Student Presentation: Solutions for Infinite Dimensional Problems

Our problem is the best convex interpolation problem, which is an infinite dimensional problem. In our problem,

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

defined by

$$x \mapsto \frac{1}{2}\langle x, x\rangle$$

is our objective. We have $\psi_i \in X = L^2[0,1]$ are the constraint functions.

Given $\hat{x} \in S$ and $\psi_1, \ldots, \psi_n$, we want to find an element $x^* \in S$ such that $\langle x^*, \psi_i\rangle = \langle \hat{x}, \psi_i\rangle$, and $x \succeq 0$. This lets us write $Ax = b$ and $x \in S$. Our minimization problem is

$$\begin{array}{cc} \min & f(x) \\ s.t. & Ax = b, x \in S \end{array}$$

We have $Q = \{x \in X : Ax = b\}$ is a closed affine variety, hence convex. S is a closed convex cone. S has empty interior (if we take a non-negative function in L^2 and we take an ϵ-interval of it, by shrinking this interval, we can get these two functions as close as we wish), and f is convex. Thus , $Q \cap S$ is a closed convex set, and F attains its minimum at a unique $\overline{x}$.

18.1 A related problem

First what we did was we looked at this problem where we drop the non-negativity constraint:

Minimize

$$f(x) = \frac{1}{2}\int_0^1 x^2(t)dt$$

over the class of functions Q (x is not neccessarily positive). So, then we have the theorem:

Theorem 18.1. *Suppose that the Gram matrix Y of the constraint functions ψ_i given by $F = \{\langle \psi_i, \psi_j \rangle\}_{i,j=1}^n$ is nonsingular. Then there is a unique minimizer $\overline{x}(t) = \sum_{i=1}^n \lambda_i \psi_i(t)$ of the functional f in the admissible set Q, where the constant vector $\lambda = (\lambda_1, \dots, \lambda_n)^T$ is given by the formula*

$$\lambda = Y^{-1} b$$

The minimal value of ...

In the proof, we use the methods of calculus of varations.

Proof. Assume that $\overline{x}$ is the minimizer of f. Construct a variation $\overline{x} + \tau v + y_1(\tau)\psi_1 + \cdots + y_n(\tau)\psi_n \in Q$ of the minimizer $\overline{x}$.

The problem is that A of the variation $\overline{x} + \epsilon v$ will not be b anymore.

We derive a necessary condition on $\overline{x}$ using the equality

$$\frac{d}{d\tau} f(...)|_{\tau=0} = 0$$

From the necessary condition, find $\overline{x}$ and λ. □

Remark: If $\overline{x}(t)$ turns out to be nonnegative almost everywhere, then it minimizes f over the set $Q \cap S$.

Now, what if we don't drop this non-negativity constraint?

Lemma 18.2. *Consider the optimization problem*

$$\begin{array}{lc} min & f(x) \\ s.t. & x \in Q_+ = \left\{ x \in L^2[0,1] : \int_0^1 \psi(t)x(t)\ dt = b, x \geq 0 \right\} \end{array}$$

with $0 \neq \psi$ *...*

18.1.1 Example: Orthonormal constraint functions

18.2 The Dual Problem

From this part, we need a constraint qualification, which is a different than the one we've seen lately. Why? Our cone's interior is empty. There are several possibilities for our CQ, one is the Borwein-Lewis CQ. In our problem, this condition is translated to asxing for a feasible x where $x > 0$ ...

The Lagrangian $L : X \times \mathbb{R}^n \times S \to \mathbb{R}$ is

$$L(x, \lambda, w) = \frac{1}{2}\langle x, x \rangle + \langle \lambda, Ax - b \rangle - \langle w, x \rangle$$

Then, our primal problem becomes

$$p^* = \min_u \max_{\lambda; w \in S} L(u, \lambda, w)$$

The dual problem is

$$d^* = \max_{\lambda; w \in S} \min_u L(u, \lambda, w)$$

Differentiating by x, we have

$$\nabla_x L(\overline{x}, \lambda, w) = \overline{x} - A^*\lambda - w = 0$$

We will derive KKT-type equations. First, primal feasibility is

$$\overline{x} = A^*\overline{\lambda} + w, w \in S$$

From dual feasibility

$$A\overline{x} = b, x \in S$$

and complementary slackness is

$$\langle w, x \rangle = 0.$$

Then, $\langle \overline{x} - A^*\overline{\lambda}, \overline{x} \rangle = 0$ therefore $\overline{x} = (A^*\overline{\lambda})_+$ is the unique solution to the problem, where $A\left((A^*\overline{\lambda})_+\right) = b$.

18.3 The way we did it

We looked at the Lagrangian

$$L(x, \lambda) = \int_0^1 \left(\frac{1}{2}|x(t)|^2 - x(t) \sum_{i=1}^n \lambda_i \psi_i(t) \right) dt + \sum_{i=1}^n \lambda_i b_i, \qquad x \in S$$

Now our dual problem is

$$d^* = \max_\lambda g(\lambda)$$

So our optimality condition is

$$\nabla_x L(\overline{x}, \lambda) \in (S - \overline{x})^+$$

where A^+ denotes the polar set of A. Equivalently,

$$\langle \nabla_x L(\overline{x}, \lambda), y - \overline{x} \rangle \geq 0, \ \forall \ y \in S.$$

Our candidate for w will be $\overline{x} - A^*\lambda$. Then, I need to show that $w \in S$ and that $\langle w, \overline{x} \rangle = 0$.

19 Portfolio Optimization

We have the problem:

$$\min \quad x^T S x \qquad \text{variance / rank of portfolio}$$

$0 \leq x \in \mathbb{R}^n$. Let x_i represent the percent of the budget that goes into asset i. We have $e^T x = 1$ and $p^T x \geq r_{min}$ (the lower bound on the expected return).

Let $x_{[i]}$ be the ordered vector by magnitude. Let

$$f_r(x) = \sum_{i=1}^{r} x_{[i]} \qquad 1 \leq r \leq n \qquad \text{sum of } r \text{ largest values.}$$

The constraint $f_r(x) \leq \alpha$ is a convex constraint since it is equivalent to the set of constraints

$$\sum_{k=1}^{r} x_{i_k} \leq \alpha, \qquad \forall\, 1 < i_1 < i_2 < \cdots < i_r \leq n.$$

If we sum up these, we have a nice collection of linear constraints, so we have a polyhedral region. In particular, this gives a convex set.

These are a lot of constraints, and we really don't want to write them all down. So we want to see if we can do something else. We consider the *knapsack problem*. If I have a vector x, and somebody says "Find the largest r components", we can think of these things with weights, and you have a knapsack. We have to decide how many of these things (up to r of them) can fit in our knapsack. We can think of the x_i's as being a value for each package:

- Consider x_i as a value for package i.
- Choose r packages to go into the knapsack.
- We want to maximize the value:

$$\begin{array}{lll} \max & x^T y & \text{(value in knapsack)} \\ s.t. & & e^T y = r \\ & & y \leq e \\ & & y \geq 0 \end{array}$$

If y ends up being a 0/1 variable, then we've solved this problem. This is a linear program. Complementary slackness tells you that each y coordinate has to be either 0 or 1 (the inequalities must hold with equality at a vertex point). Another way to see this is that the matrix defining the polyhedron is *totally unimodular*. Because each invertible determinant is ±1, this is what we divide by in our inverse matrix problem to find the coordinates of our vertex. Thus, every vertex is integral.

Let's cal this problem the primal LP. Then, the dual coefficients are going to come from the right hand side. We're going to get a minimization problem. (The dual variable to the first constraints will be t (free) and the dual variable to the second constraints will be $u \geq 0$.) The dual problem is:

$$\begin{array}{ll} \min & rt + e^T u \\ s.t. & te + Iu \geq x \\ & u \geq 0 \\ & t \text{ free} \end{array}$$

To set our α, we simply do

$$\begin{aligned} \min \quad & rt + e^T u \leq \alpha \\ s.t. \quad & te + Iu \geq x \\ & u \geq 0 \\ & t \text{ free} \end{aligned}$$

20 Student Presentation: Ill-posed Problems

Our goal is to recover the function

$$x_{exact}(t) = \begin{cases} atnohueaont \\ aeouaotnehunah \end{cases}$$

With the *kernel function*

20.1 Optimization problem

We look for the solution of $Gx = d$ with iminimal norm $N(x) = \frac{1}{2}\|x\|^2$.

From the Lagrangian

$$L(x, \lambda) = \frac{1}{2}\|x\|^2 + \lambda^T(d - Gx)$$

we get the optimality conditions:

These conditions are satisfied by

$$GG^*\lambda = d$$

and

$$x = G^*\lambda = \sum_{k=1}^{m} \lambda_k g_k$$

where

$$GG^* : \mathbb{R}^m \to \mathbb{R}^m.$$

We have reduced the inifinite dimensional optimization problem to solving a system of m linear equations:

$$GG^*\lambda = d.$$

Because the g_k are linearly independent, the GG^* is invertible and we get the unique solution $x = G^+ d$. GG^* is the $m \times m$ Gram matrix for the inner product $\langle \cdot, \cdot \rangle$ of the g_i's. Here,

$$\langle g_i, g_j \rangle = \cdots .$$

If we choose $m = 9$ and $c = 1$, we get a certain reconstruction x^*, and $\|x_{exact}\|^2 = 0.25$ and $\|x^*\|^2 = 0.22$.

20.2 Perturbed data

What if we have noise in our data? We assume we have measurement noise. In real experiments, the measurements d are usually contaminated with noise:

$$\hat{d_k} = \int_0^1 g_k(t) x_{exact}(t)\ dt + n_k, \qquad k = 1, \ldots, m$$

where we assume the n_k to be independent Gaussians.

Using the noisy data, we get a solution where $\|x^*\|^2 = 2372.9$, compared to $\|x^*\|^2 = 0.22$. This is not a very good solution, and this approach does not work with noise in our data. We have to come up with some kind of regularization so that we can still come up with some kind of reasonable solution to this data.

The reason for the bad reconstruction is due to the condition number of GG^*, specifically, 3.5176×10^{11}. This ill-conditioning results from the fact that$G$ is a compact integral operator and has unbounded inverse.

Let's assume we have a linear operator $G : X \to Y$ and we want to find a solution x to the problem

$$Gx = d.$$

Since we don't have the exact data, we're solving the problem

$$Gx = \tilde{d}.$$

Because of infeasibility, we're really going to look at the least squares

$$\|Gx - d\|$$

We look at the Moore-Penrose Inverse $x = G^+ d$. We look for the solution of $Gx = \tilde{d}$ with minimum norm. We have the singular value decomposition $\{\sigma_i, U_i, V_i\}$ of G where σ_i are the eigenvalues of GG^* (or G^*G). U_i is the orthonormal system of eigen vectors of G^*G and V_i is the orthonormal system of eigenvecotrs for GG^*. Then,

$$\begin{aligned} x &= \frac{\sum \langle \tilde{d}, v_i \rangle u_i}{\sigma_i} \\ &= \frac{\sum \langle d, v_i \rangle u_i}{\sigma_i} + \frac{\sum \langle n, v_i \rangle u_i}{\sigma_i} \end{aligned}$$

We look at the Tikhonov Regularization:

$$\begin{aligned} x &= \frac{\sigma_i \sum \langle \tilde{d}, v_i \rangle u_i}{\sigma_i^2} \\ x_\alpha &= \frac{\sigma_i \sum \langle \tilde{d}, v_i \rangle u_i}{\sigma_i^2 + \alpha^2} \end{aligned}$$

In this case, $\alpha = \frac{1}{\mu}$.

We want to minimize

$$\min \frac{1}{2}\|x\|^{+}\frac{\mu}{2}\|Gx - \hat{d}\|^2_{S^{-1}}$$

where

$$S = diag(\cdots).$$

20.3 Experiments

We solved this for several parameter values μ. How do we choose an optimal value of μ? For different μ, we either put more weight on decreasing the residue or decreasing $\|x\|$.

To understand the influence of the regularization parameter μ on the solution, we can plot a tradeoff curve

$$\left(E(x_\mu), N(x_\mu)\right).$$

How do we find the best μ? Let's minimize

$$\min_{\mu \geq 0} E(x_\mu)^2 + N(x_\mu)^2$$

We computed the optimal μ to be $\mu_{opt} = 1.5442$. This is the best we can do with our limited information. If you had *a priori* information, then maybe we could do more. But in our case, we don't know about the noise, so this is the best that we can do.

21 Student Presentation: Quadratically Constrained Quadratic Programs (QQP) and its Semidefinite Relaxation

The problem is

$$\begin{array}{lcr} \min & g_0(x) = \frac{1}{2}x^TQ_0x + b_0^Tx & \\ s.t. & \frac{1}{2}x^TQ_ix + b_i^Tx + \alpha_i \leq 0, & i = 1, \ldots, n \end{array}$$

If we have $Q_0 \succ 0$ and $Q_i \succeq 0, (i = 1, \ldots, n)$ then we know what to do. What happens when we don't have these conditions? We will form an SDP relaxation and find a lower bound to the original problem.

To reach the relaxation, we apply a trick here. Add x_0:

$$\begin{array}{lcr} \min & g_0(x) = \frac{1}{2}x^TQ_0x + b_0^Txx_0 & \\ s.t. & \frac{1}{2}x^TQ_ix + b_i^Txx_0 + \alpha_i \leq 0, & i = 1, \ldots, n \\ & x_0^2 = 1 & \end{array}$$

and a new constraint $x_0^2 = 1$. So, this has not changed anything. This is what we call the *modified original problem.* We can write this in matrices as follows:

$$\min \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 0 & \frac{1}{2}b_0^T \\ \frac{1}{2}b_0 & \frac{1}{2}Q_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix}$$

with the constraint:

$$\min \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} \alpha_i & \frac{1}{2}b_i^T \\ \frac{1}{2}b_i & \frac{1}{2}Q_i \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} \succeq 0 \tag{21}$$

and

$$\min \begin{bmatrix} x_0 \\ x \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x \end{bmatrix} = 1 \tag{22}$$

I will set up multipliers λ_i for (21) and t for (22). Then, we have the Lagrangian:

$$L(x, \lambda, t) = \lambda^T B_0 X + \sum \lambda_i X^T B_i X + t(x^T M X - 1), \tag{23}$$

where

$$B_0 = \begin{bmatrix} 0 & \frac{1}{2}b_0^T \\ \frac{1}{2}b_0 & \frac{1}{2}Q_0 \end{bmatrix}$$
$$B_i = \begin{bmatrix} \alpha_i & \frac{1}{2}b_i^T \\ \frac{1}{2}b_i & \frac{1}{2}Q_i \end{bmatrix} \qquad (i = 1, \ldots, n)$$
$$M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Thus,

$$d^* = \max_{\lambda;t} \min_{x} \left(X^T \left(B_0 + \sum \lambda_i B_I + tM \right) X \right) - t$$

So, our hidden constraint is $B_0 + \sum \lambda_i B_I + tM \succeq 0$.

We have

$$d^*_{SDP} = \max_{\lambda \geq 0; t} -t : B_0 + \sum \lambda_i B_i + tM \succeq 0, X \in \mathbf{S}^n$$

So

$$p^*_{SDP} = \min \langle B_0, X \rangle : \langle M, X \rangle = 1, \langle B_i, X \rangle \preceq 0, (i = 1, \ldots, n), X \succeq 0$$

We also know that

$$p^* \geq d^*_{SDP} = p^*_{SDP} \tag{24}$$

So, we have strong duality as long as we have a feasible point. In the first inequality of (24), we will have strong duality in the case of convexity if we have a constraint qualification.

When we have a convex program, MOSEK will give us p^* and SeDuMi will give us p^*_{SDP}, and they will be the same. When we don't have convexity, we don't get an answer from MOSEK, but we get p^*_{SDP}, a lower bound for the original problem.

22 Student Presentation: Portfolio Diversity and Robustness

Let S be the covariance matrix and r a return vector. Then, the classic model is

$$\begin{array}{cc} min & x^T S x \\ s.t. & x \geq 0, 1^T x = 1, r^T x \geq r_{min} \end{array}$$

We can solve this via the KKT conditions.

Recall the *efficient frontier*. Recall that we want to minimize the risky assets. If we fix our *stdev*, we want to maximize our mean.

Like in the Markowitz model, we talked about extensions. One of our extensions is diversity. We added diversity via a constraint. What we addressed in our project is to change that constraint into an ℓ_1 or an ℓ_2 model.

Why is diversity important? We want to diversify our risk. If we want to talk about idosyncratic risk, we think about an Enron situation. Everybody thought, because it was a blue chip stock, all employees put their money in there. But look what happened: they lost all that they have. Maybe I don't want to put all of my eggs in one basket. We want to diversify placement of our assets.

Most of our project was to look at *robustness*. This requires looking at the objective function.

Our estimator is not perfect.

Another issue is that even with a perfect estimator, why is that useful information? We're not allowed to invest in the past. How's to say that this is representative of what happens moving forward?

$$\begin{array}{cc} min & x^T S x \\ s.t. & x \geq 0, \\ & 1^T x = 1, \\ & r^T x \geq r_{min} \\ & \sum_{i=1}^{0.1n} x_{[i]} \leq \alpha \end{array}$$

Now, we think of adding an ℓ_2-norm constraint:

$$\begin{array}{cc} \min & x^T S x \\ s.t. & x \geq 0, \\ & 1^T x = 1, \\ & r^T x \geq r_{min} \\ & \sum_{i=1}^{0.1n} x_{[i]} \leq \alpha \\ & \|x\|_2 \leq u \end{array}$$

After you add ℓ_2-norm constraints, there are more coordinates that jump away from zero. This will increase the diversity of your portfolio. For different u, we can look at the number of non-zero weightings in our optimal solution. As u increases, this number decreases. After $u = 0.32$, this number is quite stationary. This is because beyond this, the $\|x\|_2 \leq u$ condition is essentially inactive. Conversely, when you decrease u, this number increases, increasing the diversity. Thus ℓ_2-norm can help us increase the diversity.

Here, we can also look at on ℓ_1-norm constraint:

$$\begin{array}{rc} \min & x^T S x \\ s.t. & -1 \leq q \leq 1 \\ & r^T x \geq r_{min} \\ & \sum_{i=1}^{0.1n} x_{[i]} \leq \alpha \\ & \|x\|_1 \leq u \end{array}$$

You might think that you'd get similar results to ℓ_2, but this is not the case. In fact, you have the exact opposite: as you increase u, the number of nonzero weightings increases, (but stabilizes to 25 after $u = 3$.

22.1 Robust Optimization

The classic model again is

$$\begin{array}{rc} min & x^T S x \\ s.t. & x \geq 0, \\ & 1^T x = 1, \\ & r^T x \geq r_{min} \end{array}$$

In robust optimization, we want to let r vary (i.e., adding infinitely many constraints). The value r is going to vary in the set E. For what kind of convex sets E can we solve this problem? If E is an ellipsoid, then we can reformulate it by this robust model:

$$\begin{array}{rc} min & x^T S x \\ s.t. & x \geq 0, \\ & 1^T x = 1, \\ & r^T x \geq r_{min} \\ & r \in E = \{\overline{r} + Pu : \|u\|_2 \leq 1\} \end{array}$$

where E is an ellipsoid.

We look at the family of constraints:

$$r^T x \geq r_{min}, r \in E = \{\overline{r} + Pu : \|u\|_2 \leq 1\}$$

We can get a new robust model.

Recall that we have our two representations for ellipsoids.

22.2 Random Matrix Theory

RMT is based on principal component analysis. The idea is to find the direction of the largest eigenvalue: this λ is interpreted as the broad market effect on the estimated S.

Our eigenvalue information is wrought from spectral decomposition. We perturb only the largest eigenvector: After we decompose S into eigenvalues and eigenvectors, we recompose S, and we take the stress matrix. We can give a random matrix reformulation.