
Applying Convex Optimisation

Koenraad M.R. Audenaert

University of Wales Bangor

November 26, 2003

Matt

- Matt is an economist and has a boring job.
- He works in a bank and manages portfolio's.
- A portfolio is a collection of assets (or stocks), held in certain amounts.
- The return of the portfolio is determined by the relative price change of every asset.
- Matt models these price changes by a random vector with given mean and covariance.
- His job is to compose an optimal portfolio:
 - starting from a given budget,
 - with a prescribed lower bound on the average return,
 - minimising the variance on the return (the risk).

Ned

- Ned has an even more boring job.
- He's a mechanical engineer, designing cantilever beams.
- These beams consist of segments with various rectangular cross-sections and fixed length.
- The left end of the beam is fixed, the right end is subject to a load.
- Ned must design these beams so as to:
 - minimise their weight
 - keeping maximal deflection and maximal stress below given limits
 - under restrictions on aspect ratios, width and height of the segments

Otis

- Otis is a really poor sod...
- He's a computer scientist.
- Otis has to write a program to minimise the travel expenses by optimising the salespersons' itineraries...
- Oh dear, oh dear! The Travelling Salesperson Problem is NP-COMPLETE.
- To find the optimal itinerary you must try out all itineraries. This might take a long time, indeed!

Koenraad

- Koenraad is a quantum mechanic.
- Koenraad works with quantum states and quantum operations, within the boundaries of quantum mechanics (Schrodinger, Heisenberg, ...)
- He has to optimise various kinds of quantum tasks, like quantum error correction, quantum teleportation, and quantum livestock management.
- He also has to prove mathematically that his solutions are optimal!

Matt, Ned, Otis and Koenraad

- What do these people have in common?
- Convex optimisation theory helps them out!

Convexity

- To explain convex optimisation, one must first explain convexity
- Sets can be convex
- Functions can be convex

Convex Sets

- A set S is **convex** iff

$$\forall p, q \in S, 0 \leq t \leq 1 : tp + (1-t)q \in S$$

or, in words, every convex combination of points in S is also in S .

- A set S is convex iff none of its tangents cuts the set.

Convex functions

- A function $f(x)$ is **convex** iff its epigraph is convex
- The **epigraph** of a function is the set

$$\text{Epi}(f) = \{(x, y) : y \geq f(x)\}.$$

- Hence,

$$\forall x_1, x_2, 0 \leq t \leq 1 : tf(x_1) + (1 - t)f(x_2) \geq f(tx_1 + (1 - t)x_2).$$

- But also: f is convex iff value $f(x)$ is never less than value of any of its tangents at x .
- Functions can be **concave**: $f(x)$ is concave iff $-f(x)$ is convex.

The Importance of Being Convex

- Convex functions have only one minimum.
- Because of that, the minimisation of convex functions does not suffer from **local minima**.
- This is also true for certain constrained minimisations of convex functions: **The minimisation of a convex function over a convex set has just one solution.**
- This is the general statement of a complex optimisation problem.
- It is nice to have a convex problem, because optimisation algorithms can't get stuck in local optima.
- But there is more: you can even find bounds on how far you are from the optimum.
- To find out how, let's have a look at the tangents!

Tangents

- Let's calculate the tangents in a simple case: a convex function $f(x)$ of 1 variable.
- Tangents are linear: $y = ax + b$, hence are determined by pairs (a, b) .
- Given the slope a , we need to calculate the offset $b = \hat{b}(a)$.
- A line is a tangent to f iff b is maximal but $ax + b$ is nowhere larger than $f(x)$.

$$\begin{aligned}\hat{b}(a) &= \max_b \{b : \forall x, ax + b \leq f(x)\} \\ &= \max_b \{b : \forall x, b \leq f(x) - ax\} \\ &= \max_b \{b : b \leq \min_x f(x) - ax\} \\ &= \min_x f(x) - ax.\end{aligned}$$

Tangents

- Tangents are linear: $y = ax + b$, hence are determined by pairs (a, b) .
- A line is a tangent to f iff b is maximal but $ax + b$ is nowhere larger than $f(x)$.

$$\hat{b}(a) = \min_x f(x) - ax.$$

- Since $f(x)$ is convex, there is only one minimum, in $x = x_0(a)$, the root of

$$f'(x) - a = 0.$$

- Thus $\hat{b}(a) = f(x_0(a)) - ax_0(a)$.
- The function $\hat{b}(a)$ is concave in a , since it is the pointwise minimum of functions $f(x) - ax$, linear in a .

A Simple Convex Problem

- Let's tackle the following constrained optimisation:
- Minimise a convex function $f(x)$ over the interval $x \leq x_u$.

$$\hat{f} = \min_x \{f(x) : x \leq x_u\}.$$

- We can easily prove

$$\min_x \{f(x) : x \leq x_u\} \geq \max_a \{ax_u + \hat{b}(a) : a \leq 0\}.$$

(In fact, equality holds!)

- The maximisation over a is a convex problem, called the **dual** problem.
- The minimisation over x is called the **primal** problem.

Certificates of Convergence

- By picking specific $x \leq x_u$ (so-called **feasible** x), you get upper bounds on \hat{f} .
- By picking specific tangents $a \leq 0$, you get lower bounds on \hat{f} .
- Thus if you solve the primal problem together with the dual one, you can bracket the solution within an upper and a lower bound.
- The difference between the bounds is an upper bound on how far you are from the real solution.
- You, therefore, get a **certificate of convergence**.
- This works for general convex optimisation problems.
- No other optimisation problem has this feature.

A Proof

- We have found earlier $\hat{b}(a) = \min_x f(x) - ax$.
- $= \min_x f(x) - ax_u + a(x_u - x)$.
- Under the constraints $a \leq 0, x \leq x_u$, we have $a(x_u - x) \leq 0$,
- so that $\hat{b}(a) \leq \min_x f(x) - ax_u$, or
- $f(x) \geq ax_u + \hat{b}(a)$ whenever $a \leq 0, x \leq x_u$.
- If we now minimise the LHS over the feasible x , and maximise the RHS over the feasible a , we get

$$\min_x \{f(x) : x \leq x_u\} \geq \max_a \{ax_u + \hat{b}(a) : a \leq 0\}.$$

QED

Classes of Convex Problems

- **Linear programming**: target function linear; constraints are linear (in)-equalities.
- **Quadratic Programming**: target function quadratic; constraints are linear (in)-equalities.
- **Geometric Programming**: target function posynomial; constraints are posynomial (in)-equalities.
- **Semidefinite Programming**: minimise a function linear in a vector x , subject to a linear matrix inequality (LMI) $F_0 + x_1F_1 + x_2F_2 + \dots \geq 0$, i.e. the resulting matrix is positive semidefinite (PSD). This is equivalent to minimising a linear function over the set of PSD matrices, subject to linear equalities.

Numerically Solving Convex Problems

- A lot of very efficient algorithms have been devised.
 - Cutting-Plane Methods
 - Barrier Methods
 - Primal-Dual Methods
- Libraries in C/C++, Fortran
- Matlab Toolboxes: SeDuMi (for free), Mosek (not for free, but well...)
- Some excellent references, containing information about algorithms and applications:
 - S. Boyd and L. Vandenberghe, *Convex Optimization*, available online at <http://www.stanford.edu/~boyd/cvxbook.html> (2002).
 - L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, March 1996.

Approximating NP-Complete Problems

- Quite surprisingly, semi-definite programming can help in finding “good” solutions to several NP-complete problems.
- We’ll take as an example MAX-CUT.
- “Given a weighted graph with n vertices and edge weights w_{ij} , find a partitioning of the graph into two parts such that the sum of the weights of the edges cut by this partitioning is maximal.”
- Let the variable x_i be $+1$ if vertex i is in one part, and -1 if it is in the other, then
- MAX-CUT is the maximisation

$$W_{\text{opt}} = \max_{x \in \{-1, +1\}^n} \sum_{i < j} w_{ij} (1 - x_i x_j) / 2.$$

A randomised MAX-CUT algorithm

- Since MAX-CUT is NP-complete, people have been looking at finding simple algorithms that perform “well”.
- “Well” means: yielding a suboptimal solution with a guaranteed performance.
- Performance = value of solution/value of optimum
- Randomised algorithm for MAX-CUT: choose the x_i at random (uniformly).
- One can show: average performance is at least 50%.
- This has been the best randomised algorithm for nearly two decades.
- 1994: Goemans and Williamson: a larger hammer.

A better MAX-CUT

- Basic idea: introduce correlations between the chosen x_i .
- To every vertex $i \in \{1, \dots, n\}$ associate a (judiciously chosen) unit vector $v_i \in \mathbb{R}^n$.
- Generate a random vector $r \in \mathbb{R}^n$.
- Let $x_i = \text{sgn}\langle v_i, r \rangle$.
- How to choose the unit vectors v_i ? Optimise average performance!
- The outcome of a run is

$$W = \sum_{i < j} w_{ij} (1 - \delta(\text{sgn}\langle v_i, r \rangle, \text{sgn}\langle v_j, r \rangle)).$$

- The average outcome is thus

$$E[W] = \sum_{i < j} w_{ij} P(\text{sgn}\langle v_i, r \rangle \neq \text{sgn}\langle v_j, r \rangle).$$

A better MAX-CUT

- The average outcome is

$$E[W] = \sum_{i < j} w_{ij} P(\text{sgn}\langle v_i, r \rangle \neq \text{sgn}\langle v_j, r \rangle).$$

- A simple calculation for the probability reveals

$$E[W] = \sum_{i < j} w_{ij} \arccos(\langle v_i, v_j \rangle) / \pi.$$

- A linear lower bound for the arccos gives

$$E[W] = \alpha \sum_{i < j} w_{ij} (1 - \langle v_i, v_j \rangle) / 2,$$

where $\alpha = 0.87856$.

A better MAX-CUT

- Look at the RHS of

$$E[W] = \alpha \sum_{i < j} w_{ij} (1 - \langle v_i, v_j \rangle) / 2,$$

- That looks much like the original MAX-CUT problem, but with every x_i (being ± 1) replaced by an n -dimensional unit vector v_i .
- But this is actually a relaxation of MAX-CUT, so the optimal value of the sum will be $\geq W_{\text{opt}}$.
- Hence $E[W] \geq \alpha W_{\text{opt}}$, i.e. the average performance is $\alpha = 0.87856$.

A better MAX-CUT

- This better performance will be achieved for those v_i optimising the relaxed problem

$$W_{\text{rel}} = \max_{v_i \in \mathbb{R}^n} \left\{ \sum_{i < j} w_{ij} (1 - \langle v_i, v_j \rangle) / 2 : \|v_i\| = 1 \right\}.$$

- This can be recast as an SDP problem!
- Introduce the Gram matrix Y : $Y_{ij} = \langle v_i, v_j \rangle$. It is a fact of life that a matrix is a Gram matrix if and only if it is PSD. Hence

$$W_{\text{rel}} = \max_{Y \geq 0} \left\{ \sum_{i < j} w_{ij} (1 - Y_{ij}) / 2 : Y_{ii} = 1 \right\}.$$

- Indeed, a maximisation of a linear function of a PSD matrix subject to linear equalities.

An even better MAX-CUT?

- A better MAX-CUT was found by introducing correlations between the random x_i .
- Question to the Quants: can we find an even better MAX-CUT by introducing **quantum** correlations?
- Choose an n -qubit state ψ and measure every qubit in, say, the z -basis, yielding values for x_i .
- QM tells us that with an entangled state ψ we can sometimes find stronger correlations than in the classical case.
- To be investigated...

Other Applications

- Matt's problem, **Markowitz Portfolio Optimisation**, is a quadratic programming problem:

$$\min_x \{ \langle x, \Sigma x \rangle : \langle E[p], x \rangle \geq r_{\min}, \sum_i x_i = B, x_i \geq 0 \}$$

where

- x_i are the assets,
- $E[p_i]$ is the expected relative price change of asset i ,
- Σ is the covariance matrix of these price changes,
- r_{\min} is the minimum desired return of the portfolio,
- and B is the budget.

Other Applications

- Ned's problem, **Structural Optimisation**, is a geometric program:
- minimise weight, proportional to $\sum_i w_i h_i$, subject to the constraints:
 - $w_{\min} \leq w_i \leq w_{\max}, h_{\min} \leq h_i \leq h_{\max}$
 - $a_{\min} \leq h_i/w_i \leq a_{\max}$
 - maximal stress: $6iF/(w_i h_i^2) \leq \sigma_{\max}$
 - maximal deflection at end: $y_1 \leq y_{\max}$
- here, y_1 is to be calculated from a recursion formula involving deflections y_i and slopes v_i of the right ends of the beam segments:

$$v_i = 12(i - 1/2)F/Ew_i h_i^3 + v_{i+1}$$

$$y_i = 6(i - 1/3)F/Ew_i h_i^3 + v_{i+1} + y_{i+1}$$

$$v_{n+1} = y_{n+1} = 0.$$

Other Applications

- Many of Koenraad's problems in Quantum Information Theory are Semidefinite Programs!
- In Quantum Mechanics, PSD matrices are used all over the place:
 - **States** are PSD matrices, because the eigenvalues of states are probabilities and must therefore be real and positive.
 - **Operations** are PSD matrices, because QM imposes all operations to be linear, and they must map PSD matrices to PSD matrices (states to states).
- A lot of problems in QIT involve optimisation: find the best error-correcting operation,...
- When using linear measures of goodness these optimisations are SDP problems.

Conclusion: Take-home message

- Convex Programming is useful in a surprising number of applications, in a variety of domains.
- When your optimisation problem turns out to be convex you should be forever grateful, because you can **solve it!**
- Efficient numerical software exists.
- No local optima, no swamping.
- Using duality, you get certificates of convergence.
- Sometimes an analytical proof of optimality is possible.