Step1_ConvexProgramming

March 11, 2020

[ ]:
```
using Convex
using GLPK
using Printf
```

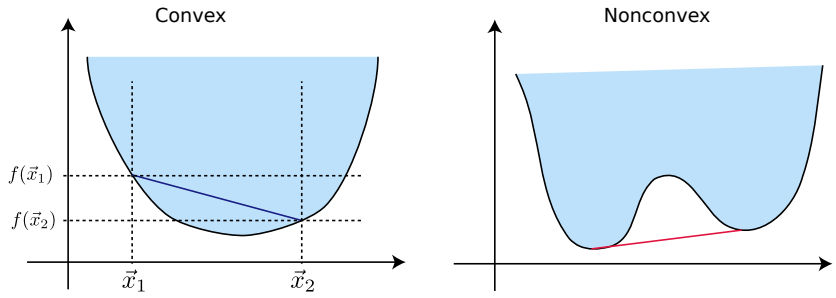
1 Introduction to convex optimization

$$\begin{array}{ll} \text{minimize} & f(\vec{x}) \\ \text{over} & \vec{x} \in R^n \\ & g_j(\vec{x}) \leq 0 \quad j \in \mathcal{J} \\ & h_k(\vec{x}) = 0 \quad k \in \mathcal{K} \end{array} \tag{1}$$

where all f, g_j and h_k are convex; for $f(\vec{x})$:

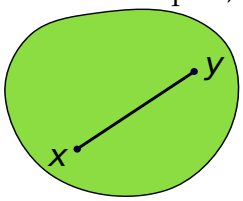
$$f(\alpha\vec{x}_1 + (1-\alpha)\vec{x}_2) \leqslant \alpha f(\vec{x}_1) + (1-\alpha)f(\vec{x}_2), \qquad \alpha \in [0,1] \tag{2}$$

On left, an example of a convex function is given below, where we plot the line segment $\alpha f(\vec{x}_1) + (1-\alpha)f(\vec{x}_2)$ in dark blue. On the right, an example of a nonconvex function.

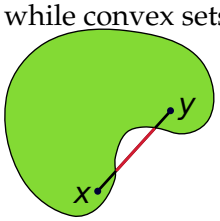


ConvexFun.svg

In the same spirit, convex sets include the line segment between any two of their points:



while convex sets do not:



Convex programs (1) have one crucial property: any local minimum is also a global minimum. In particular, this guarantees that (non pathological) optimization algorithms always find the global minimum up to numerical precision.

1.1 Linear optimization

The simplest kind of convex programs is a *linear program*, where the objective and all constraints are represented by linear functions such as (technically, these are *affine functions*):

$$f(\vec{x}) = \vec{c}^\top \vec{x} + d \tag{3}$$

$$\begin{aligned} p^\star = \text{minimize} \quad & x_1 + 2x_2 - x_3 \\ \text{over} \quad & \vec{x} \in R^3 \\ & x_1, x_2, x_3 \geq 0 \\ & x_1 + x_2 + x_3 = 1 \end{aligned} \tag{4}$$

We now solve that problem using the Convex.jl optimization toolbox.

[ ]:
```
x = Variable(3)
constraints = [x >= 0, sum(x) == 1]
objective = x[1] + 2*x[2] - x[3]
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL ))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point     x = %s\n", x.value)
```

1.1.1 Primal canonical form of a linear program

The standard form of a linear program is as follows, with $\vec{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^m$.

$$\begin{aligned} \text{minimize} \quad & \vec{c}^\top \vec{x} \\ \text{over} \quad & \vec{x} \in R^n \\ & A\vec{x} = \vec{b} \\ & \vec{x} \geq 0 \end{aligned} \tag{5}$$

The problem is fully specified by the matrix A and the vectors $\vec{b}$ and $\vec{c}$.

Lecture exercice Complete the program data below to solve (4).

[ ]:
```
using Convex
using GLPK
# remember [1, 1, 1] is a 1D vector, [1; 1; 1] is a column matrix, [1 1 1] is a␣
 ↪row matrix
```

```
x = Variable(3)
c = []
A = []
b = []
constraints = [A*x == b, x >= 0]
objective = dot(c, x)
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective p = %f\n", problem.optval)
@printf("Optimal point     x = %s\n", x.value)
```

1.2 Duality

We remind the primal canonical form (5):

$$p^\star = \underset{\vec{x}\in R^n}{\text{minimize}} \quad \vec{c}^\top \vec{x}, \qquad A\vec{x} = \vec{b}, \qquad \vec{x} \geq 0,$$

and introduce the notation

- a variable with a $\star$ superscript is an optimal solution (as in $p^\star$ is the minimum of the LP),
- a variable with a $*$ superscript is a feasible solution (as in $\vec{x}^*$ satisfies the constraints),

with the conventions

- $p^\star = +\infty$ if the problem is infeasible,
- $p^\star$ is finite if the problem has an optimal (thus feasible) solution,
- $p^\star = -\infty$ if the problem is unbounded, as there are feasible solutions with $\vec{c}^\top \vec{x} \to -\infty$.

We introduce the Lagrange multipliers $\vec{y} \in \mathbb{R}^m$ corresponding to the constraint $A\vec{x} = \vec{b}$, and the Lagrange multipliers $\vec{s} \in \mathbb{R}^n$ corresponding to the constraint $\vec{x} \geq 0$, with $\vec{s} \geq 0$ as they correspond to an inequality constraint. The Lagrangian function is:

$$L(\vec{x}, \vec{y}, \vec{s}) = \vec{c}^\top x + \vec{y}^\top (\vec{b} - A\vec{x}) - \vec{s}^\top \vec{x}. \tag{6}$$

For any feasible $\vec{x}^*$ and any $(\vec{y}^*, \vec{s}^*)$ with $\vec{s} \geq 0$, we have

$$L(\vec{x}^*, \vec{y}^*, \vec{s}^*) = \vec{c}^\top x^* + (\vec{y}^*)^\top \underbrace{(\vec{b} - A\vec{x}^*)}_{=0} - (\vec{s}^*)^\top \vec{x}^* \leq \vec{c}^\top \vec{x}^* \tag{7}$$

Let us now minimize $L(\vec{x}, \vec{y}, \vec{s})$ over $\vec{x}$:

$$g(\vec{y}, \vec{s}) = \min_{\vec{x}} \vec{x}^\top (\vec{c} - A^\top \vec{y} - \vec{s}) + \vec{b}^\top \vec{y} = \begin{cases} \vec{b}^\top \vec{y}, & \vec{c} - A^\top \vec{y} - \vec{s} = 0, \\ -\infty, & \text{otherwise.} \end{cases} \tag{8}$$

Note that for every $(\vec{y}, \vec{s})$ (with $\vec{s} \geq 0$ by definition), the value $g(\vec{y}, \vec{s})$ is a lower bound for $p^\star$ by (7).

In (8), only the first case ($\vec{c} - A^\top \vec{y} - \vec{s} = 0$) leads to a nontrivial upper bound. We now maximize over such $(\vec{y}, \vec{s})$ to get the best lower bound.

1.2.1 Dual canonical form of a linear program

We obtain:

$$\begin{array}{rll} d^\star = \text{maximize} & \vec{b}^\top \vec{y} & \\ \text{over} & \vec{y} \in \mathbb{R}^m, \vec{s} \in \mathbb{S}^n & \\ & \vec{c} - A^\top \vec{y} = \vec{s} & \\ & \vec{s} \geq 0 & \end{array} \tag{9}$$

which is the dual problem. Through the Lagrangian mechanism, we have $d^\star \leq p^\star$ which is *weak duality*.

We have the cases, for the dual problem: - $d^\star = +\infty$, dual unbounded, thus $+\infty \leq p^\star$ and the primal problem is infeasible, - $d^\star$ is finite, the dual is feasible, the primal problem can either be feasible or infeasible (but not unbounded), - $d^\star = -\infty$, the dual problem is infeasible.

For linear programs only, when $d^\star$ is finite, then $p^\star = d^\star$; when $p^\star$ is finite, also $d^\star = p^\star$ (this is called *strong duality*).

Lecture exercice Fill the problem data below to solve the dual of (4).

[ ]:
```
using Convex
using GLPK
y = Variable(1)
s = Variable(3)
c = []
A = []
b = []
constraints = [s >= 0, s == c - transpose(A)*y]
objective = dot(b, y)
problem = Convex.maximize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL))
@printf("\n")
@printf("Optimal objective d = %f\n", problem.optval)
@printf("Optimal point     y = %s\n", y.value)
@printf("Optimal point     s = %s\n", s.value)
```

1.2.2 Recovering dual variables using Convex.jl

Most convex solvers solve the primal and the dual problem at the same time, as doing so lead to better robustness (the primal-dual gap $p^\star - d^\star$ is then a measure of convergence).

Remember the primal problem:

$$p^\star = \underset{\vec{x} \in R^n}{\text{minimize}} \quad \vec{c}^\top \vec{x}, \qquad A\vec{x} = \vec{b}, \qquad \vec{x} \geq 0,$$

and realize that any feasible $\vec{x}^*$ provides an objective p^* that is an upper bound on the true minimum by definition: $p^\star \leq p^*$.

The same happens for the dual problem: any feasible pair $(\vec{y}^*, \vec{s}^*)$ provides an objective d^* that is a lower bound on the true dual maximum.

Thus if we have a primal feasible solution $\vec{x}^*$ and a dual feasible solution $(\vec{y}^*, \vec{s}^*)$, we sandwich the true solution $d^\star = p^\star$:

$$d^* \leq d^\star = p^\star \leq p^*. \tag{10}$$

Linear programming solvers try to return a feasible solution pair for both the primal and dual, with primal objective $\tilde{p}^*$ and dual objective $\tilde{d}^*$. It would be tempting to imagine that the true solution lies in the internal $[\tilde{d}^*, \tilde{p}^*]$. However the primal and dual solutions are usually *slightly* infeasible due to numerical errors, and the relation (10) only holds approximately. Worse, it's difficult to assess the impact of such numerical errors just by looking at the problem data.

Below, we provide the primal problem to Convex.jl, and recover the dual variables from the solution. Compare and contrast with the dual formulation above.

[ ]:
```
using Convex
using GLPK
x = Variable(3)
c = []
A = []
b = []
constraints = [A*x == b, x >= 0]
objective = dot(c, x)
problem = Convex.minimize(objective, constraints)
solve!(problem, GLPK.Optimizer(msg_lev=GLPK.MSG_ALL ))
@printf("\n")
@printf("Optimal objective d = %f\n", problem.optval)
@printf("Optimal point     x = %s\n", x.value)
@printf("Dual variable 1   %s\n", problem.constraints[1].dual)
@printf("Dual variable 2   %s\n", problem.constraints[2].dual)
```

[ ]:
```

```