Problem 2.1: Convexity Properties of Sets

(a) Let $\{S_i\}_{i\in M}$ be a collection of $M = \{1,\dots,m\}$ convex sets in $\mathbb{R}^n$. Show that their intersection $S = \cap_{i\in M} S_i$ is also convex.

(b) Let S_1 and S_2 be closed convex sets in $\mathbb{R}^n$. Show that their Minkowski sum

$$S = S_1 + S_2 = \{x + y : x \in S_1, y \in S_2\}$$

is also convex. Also, show by an example that $S_1 + S_2$ is not necessarily closed.

Solution.

(a) Let $x, y \in S$ and $0 \le \lambda \le 1$. By the definition of S, we must have $x, y \in S_i$ for all $i \in M$. Since each S_i is convex, we must also have $\lambda x + (1-\lambda)y \in S_i$ for all $i \in M$. Therefore, $\lambda x + (1-\lambda)y \in \cap_{i\in M} S_i = S$, and thus S is also convex (as we selected $x, y \in S$ randomly).

(b) Let $x_1, x_2 \in S_1$ and $y_1, y_2 \in S_2$. Thus, we have $x_1 + y_1 \in S_1 + S_2$ and $x_2 + y_2 \in S_1 + S_2$. Letting $0 \le \lambda \le 1$ and applying the definition of convexity, we get

$$\begin{aligned}\lambda(x_1+y_1) + (1-\lambda)(x_2+y_2) &= \lambda x_1 + \lambda y_1 + x_2 + y_2 - \lambda x_2 - \lambda y_2 \\ &= \underbrace{\lambda x_1 + (1-\lambda)x_2}_{\in S_1} + \underbrace{\lambda x_1 + (1-\lambda)x_2}_{\in S_2} \in S_1 + S_2 = S\end{aligned}$$

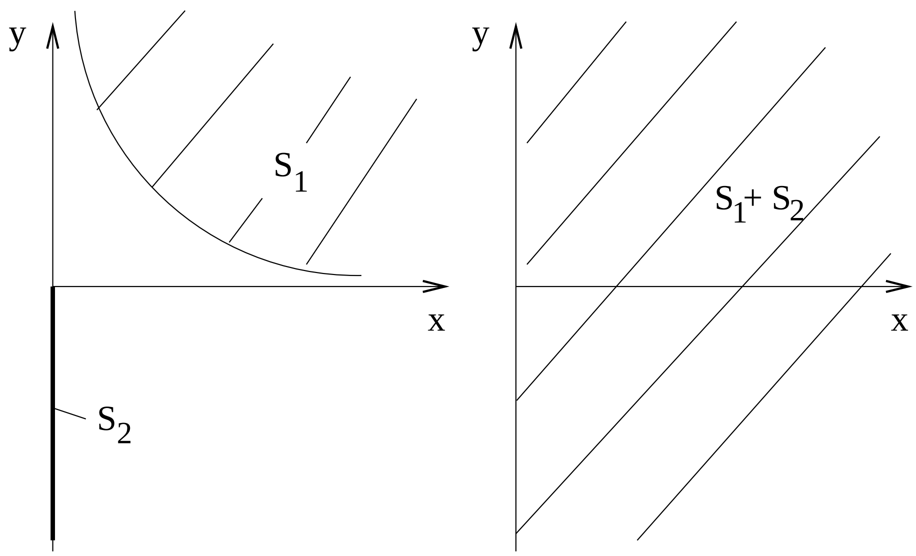
Since a convex combination of any two points $(x_1 + y_1) \in S$ and $(x_2 + y_2) \in S$ belongs to $S = S_1 + S_2$, the set S must be convex.

Next, let us show by example that $S_1 + S_2$ is not necessarily closed even though S_1 and S_2 are closed. Let S_1 and S_2 be the following closed, convex sets:

$$S_1 = \{(x, y) \in \mathbf{R}^2 \mid y \ge 1/x, x > 0\}$$

$$S_2 = \{(x, y) \in \mathbf{R}^2 \mid x = 0, y \le 0\}$$

Their Minkowski sum $S = S_1 + S_2 = \{(x, y) \in \mathbb{R}^2 : x > 0, y \in \mathbb{R}\}$ is neither open nor closed.



Problem 2.2: Weierstrass' Theorem

Consider the following nonlinear optimisation problem P:

$$\begin{aligned} (P):\ \max_{x,y}.\ & \frac{1}{x+y} \\ \text{subject to: } & xy \geq 1 \\ & x, y \geq 0 \end{aligned}$$

(a) Show that P has a solution by applying Weierstrass' theorem.

(b) Model the problem P with `JuMP` and try to find the global maximum.

Solution.

(a) The Weierstrass' theorem is the following:

Theorem 1 (Weierstrass' theorem) *Let $S \neq \emptyset$ be a compact set, and let $f : S \to \mathbb{R}$ be continuous on S. Then there is a maximizing solution to*

$$(P):\ z = max.\ \{f(x) : x \in S\}.$$

Now we have

$$f(x, y) = \frac{1}{x+y} \quad \text{and} \quad S = \{(x, y) \in \mathbb{R}^2 : xy \geq 1, x \geq 0, y \geq 0\}$$

The function $f(x, y)$ is continuous on S, but the feasible set S is not bounded and therefore not compact. However, we can partition S into two parts, for example:

$$S = S_1 \cup S_2 = \underbrace{\{S : x + y \leq 6\}}_{S_1} \cup \underbrace{\{S : x + y \geq 6\}}_{S_2}$$

S_1 is closed and bounded and therefore compact, whereas S_2 is closed but not bounded (and thus not compact). Now from the definitions of S_1 and S_2, we get the following bounds

$$\begin{aligned} f(x, y) &= \frac{1}{x+y} \geq \frac{1}{6}, \text{ for all } (x, y) \in S_1 \\ f(x, y) &= \frac{1}{x+y} \leq \frac{1}{6}, \text{ for all } (x, y) \in S_2 \end{aligned}$$

Thus, the optimal solution will be part of the set S_1 since it always produces greater than or equal objective function values than solutions in set S_2, and we can focus solely on S_1.

Now, as $f(x, y)$ is continuous in S_1 and S_1 is compact (i.e., closed and bounded), Weierstrass' theorem guarantees that the problem has a maximizing solution.

(b) In this case, the maximizing solution is $(x, y) = (1, 1)$ with $f(x, y) = 0.5$. See the Julia code which solves the optimization problem.

Problem 2.3: Portfolio Optimization

For this problem, use the data file prices.csv which contains daily prices of $N = \{1, \ldots, n\}$ stocks over a time period of $T = \{1, \ldots, m\}$ days. Let $x_i \geq 0$ denote the (long) position of stock $i \in N$ in a portfolio throughout the time period. The positions $x = (x_1, \ldots, x_n)$ in the portfolio are scaled to represent fractions of the total investment, that is,

$$\sum_{i \in N} x_i = 1$$

Let p_i^t denote the daily price of stock $i \in N$ for all $t \in T$, and let r_i^t be the relative daily return of stock $i \in N$ for all $t \in T \setminus \{m\}$. These are computed as

$$r_i^t = \frac{p_i^{t+1} - p_i^t}{p_i^t}, \quad \forall i \in N, \forall t \in T \setminus \{m\}$$

Let $\mu = (\mu_1, \ldots, \mu_n)$ denote the *expected relative returns* of the stocks N, and let $\Sigma \in \mathbb{R}^{n \times n}$ be the corresponding covariance matrix. Thus, the expected average return and variance of a portfolio $x = (x_1, \ldots, x_n)$ are $\mu^\top x$ and $x^\top \Sigma x$, respectively. Moreover, let $\sigma \in \mathbb{R}^n$ be the standard deviation vector and $\rho \in \mathbb{R}^{n \times n}$ the correlation matrix of the relative stock returns.

(a) Read the data and plot the price curves of each stock for the whole time period.

(b) Compute the expected average returns μ, the covariance matrix Σ, the correlation matrix ρ, and the standard deviation vector σ using the `Julia` package `Statistics`.

(c) Sort the stocks in increasing order with respect to their expected returns. Using this order, plot the expected returns μ_i and standard deviations σ_i of each stock $i \in N$ in two different plots but in the same figure. Look at Exercise 1.1 code for reference how to plot multiple plots in the same figure using the `Plots` package. **Note:** plots might not appear in Jupyter notebooks unless they are called at the last line of a cell. However, you can always save the most recent plot as a `pdf` file, for example, by calling the function `savefig("myplot.pdf")`.

(d) Using the same order as in (c), visualize the correlation matrix ρ using the `PyPlot` package function `imshow`, and make a `scatter` plot of the the stocks' expected returns vs. their standard deviations, i.e., plot the points (σ_i, μ_i), for all $i \in N$. **Note:** to save the correlation plot as a `pdf` file, you have to call `PyPlot.savefig("corrplot.pdf")` explicitly so that `Julia` knows which plotting library was used. This is needed because `Plots` and `PyPlot` both define this function with identical name and parameter types.

(e) Consider the following portfolio optimization problem

$$\begin{aligned} \min_{x}. \quad & x^\top \Sigma x && (1) \\ \text{subject to:} \quad & \mu^\top x \geq \mu_{min} && (2) \\ & \sum_{i \in N} x_i = 1 && (3) \\ & x \geq 0 && (4) \end{aligned}$$

where the objective is to minimise the portfolio variance (i.e., risk) $x^\top \Sigma x$ by satisfying a minimum expected return constraint (2). Model the problem (1) – (4) using `JuMP` and solve the problem with different values of μ_{min}. Use the `Plots` function `bar` to plot fractions of capital invested in each stock in the resulting portfolio. You can try values of μ_{min} between

$$0 \leq \mu_{min} \leq 0.000869.$$

(f) Compute the optimal portfolio with 50 different values of μ_{min} between $[0, 0.000869]$ and plot the optimal trade-off curve, i.e., the expected returns or each portfolio as a function of their standard deviations. Also, plot the points (σ_i, μ_i), for all $i \in N$, in the same figure for comparison using the function `scatter!` from the `Plots` package.

Solution

See the Julia code. Note that we used a simplification in the computations where we don't allow short positions for the stocks, so the minimum risk portfolio obtained with $\mu_{min} = 0$, for example, does not correspond to investing all capital to the lowest risk stock as one would expect. For a more realistic case, there is a meaningful connection between the covariance matrix Σ and its eigenvalues and eigenvectors. If you want to find out more information about this topic, here is a one reference.