Optimization: Part 2

Abhinav Anand, IIMB

2019/07/01

Background

The problems seeking to maximize profits or minimize costs often feature nontrivial constraints which the optimal needs to satisfy. The solution must lie at the intersection of constraints (for equality constraints) or on one side of the constraint surface (for inequality constrints).

The problem in a general form is:

$$\max f(x) : x \in \mathbb{R}^n, x \geq 0$$

$$g_1(x_1, \ldots, x_n) \leq b_1, \ldots, g_k(x_1, \ldots, x_n) \leq b_k$$

$$h_1(x_1, \ldots, x_n) = c_1, \ldots, h_m(x_1, \ldots, x_n) = c_m$$

The objective function f is real valued, i.e., $f : \mathbb{R}^n \to \mathbb{R}$; $g(\cdot)$ are functional forms of the *inequality* constraints while $h(\cdot)$ are functional forms for the *equality* constraints.

Equality Constraints

Consider the case when $x = (x_1, x_2)$ and there is a single equality constraint $h_1(x) = c_1$.

$$\max f(x) : x \in \mathbb{R}^2, x \geq 0$$

$$h_1(x) = c_1$$

To make the illustration more concrete, consider $f(x) = x_1 x_2$ and $h_1(x) = c_1 : x_1 + x_2 = 5$.

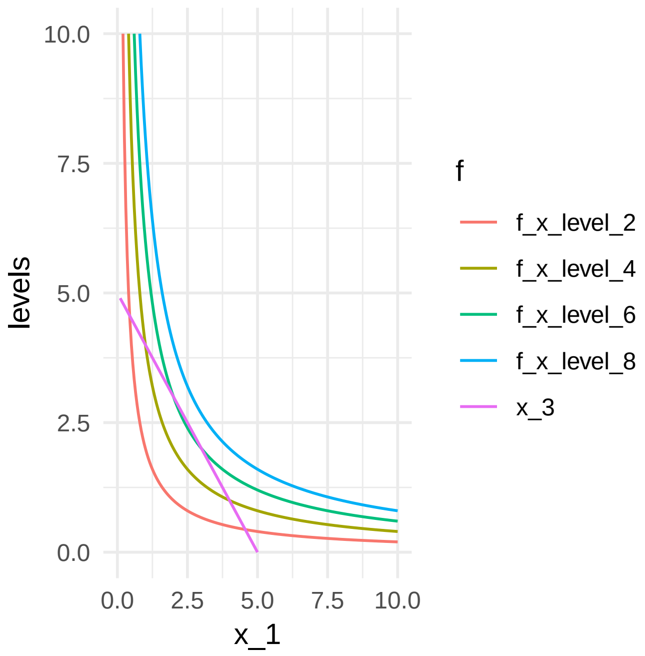
```r
x_1 <- seq(0.1, 10, 0.05)
x_2 <- seq(0.1, 10, 0.05)

f_x_level_2 <- 2/x_2 #level sets
f_x_level_4 <- 4/x_2
f_x_level_6 <- 6/x_2
f_x_level_8 <- 8/x_2

x_3 <- 5 - x_2 #constraint set

data_obj_l <- cbind(x_1,
                    f_x_level_2,
                    f_x_level_4,
                    f_x_level_6,
                    f_x_level_8,
                    x_3
                    ) %>%
  dplyr::as_tibble() %>%
  tidyr::gather(.,
                f_x_level_2:x_3,
                key = "f",
                value = "levels")

ggplot(data_obj_l, aes(x_1, levels, color = f)) +
  geom_line() +
  scale_y_continuous(limits = c(0, 10)) +
  scale_x_continuous(limits = c(0, 10)) +
  theme_minimal()
```



Geometrically we need to find the highest valued level set for $f(x) = x_1x_2$ that satisfies $x_1 + x_2 = 5, x_1, x_2 \geq 0$. The key observation is the following: at the optimal, the levels sets and the constraint set must be tangent—just touching (intersecting) each other at exactly one point. (Why must this be so? What happens if the plots cross over? Can we improve the objective function then?)

The Lagrangian

If the curves are tangents to each other at the optimal point then it must be so that the tangents to the curves at the optimal are in the same direction.

The slope of the level set of f at x^* is:

$$-\frac{\frac{\partial f}{\partial x_1}(x^*)}{\frac{\partial f}{\partial x_2}(x^*)}$$

and that of the equality constraint is:

$$-\frac{\frac{\partial h}{\partial x_1}(x^*)}{\frac{\partial h}{\partial x_2}(x^*)}$$

Since they're equal

$$\frac{\frac{\partial f}{\partial x_1}(x^*)}{\frac{\partial f}{\partial x_2}(x^*)} = \frac{\frac{\partial h}{\partial x_1}(x^*)}{\frac{\partial h}{\partial x_2}(x^*)} = \lambda$$

This can be rearranged as:

$$\frac{\frac{\partial f}{\partial x_1}(x^*)}{\frac{\partial h}{\partial x_1}(x^*)} = \frac{\frac{\partial f}{\partial x_2}(x^*)}{\frac{\partial h}{\partial x_2}(x^*)} = \lambda$$

or,

$$\frac{\partial f}{\partial x_1}(x^*) - \lambda \frac{\partial h}{\partial x_1}(x^*) = 0$$
$$\frac{\partial f}{\partial x_2}(x^*) - \lambda \frac{\partial h}{\partial x_2}(x^*) = 0$$

There are three unknowns:(x_1^*, x_2^*, λ). There are two equations above, and there is a third equation—the constraint equation $h(x_1, x_2) = c_1$. Together, we can find x^* and λ.

The following function is formally referred to as the *Lagrangian*, and λ as the Lagrange multiplier:

$$\mathcal{L}(x_1, x_2, \lambda) := f(x_1, x_2) - \lambda \cdot (h(x_1, x_2) - c)$$

We consider the critical points of the Lagrangian: $\frac{\partial \mathcal{L}}{\partial x_1}(x^*), \frac{\partial \mathcal{L}}{\partial x_2}(x^*), \frac{\partial \mathcal{L}}{\partial \lambda}(x^*) = 0$.

Essentially by forming the Lagrangian, we are transforming a constrained optimization program featuring the objective function $f(\cdot)$ into an *unconstrained* optimization program featuring the Lagrangian $\mathcal{L}(\cdot)$. However, there is an extra variable λ, the Lagrange multiplier, that is introduced in the new program.

Constraint Qualification

In order for the slopes to be well-defined, $\frac{\partial h}{\partial x_1}(x^*) \neq 0$ and $\frac{\partial h}{\partial x_2}(x^*) \neq 0$. Since this is a restriction on the constraint set, it's called a *constraint qualification.*

Theorem: For the function $f : \mathbb{R}^n \to \mathbb{R}, f \in C^1$, if $x^* \in \mathbb{R}^n$ is a solution of

$$\max f(x) : x \geq 0, h(x) = c_1$$

and x^* is *not* a critical point of h; then, there is a real number λ^* such that (x^*, λ^*) is a critical point of $\mathcal{L} = f(x) - \lambda \cdot (h(x) - c)$.

Gradients and Level-Sets

Directional Derivatives and Gradients

When the domain of the function f is one: $f : \mathbb{R} \to \mathbb{R}$ the idea of a directional derivative coincides with that of a derivative. In more than one dimension, however, the distinction is nontrivial.

Directional derivatives measure the rate of change of a function f in the direction v. It's defined as:

$$Df_v(x) = \lim_{t \to 0} \frac{f(x+tv) - f(x)}{t}$$

Using the L'Hopital and chain rule:

$$\begin{aligned} Df_v(x) &= f'(x) \cdot v \\ Df_v(x) &= \left[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right] \cdot v \end{aligned}$$

The expression $[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)]$ is the generalization of the idea of a derivative of a real valued function to n dimensions.

$$\nabla f_v(x) = f'(x) \cdot v = \left[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right] \cdot v = (\nabla f(x))^\top v$$

$$D_v f = \nabla f_v(x) = (\nabla f)^\top v$$

Since the above is an inner product, it takes its maximum value when v and ∇f are in the *same* direction. Hence we can make the claim that the gradient is in the direction of fastest increase of its function.

Additionally, all directions v such that $(\nabla f)^\top v = 0$ imply that the gradient points in a direction orthogonal to v.

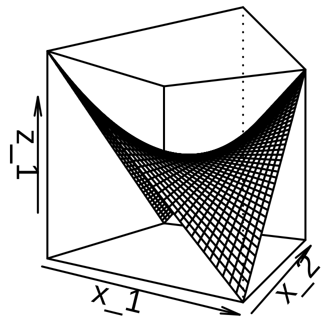
Level Sets

For a general function $f : \mathbb{R}^n \to \mathbb{R}$ a *level-set* is the set of points $x \in \mathbb{R}^n : f(x_1, \ldots, x_n) = b$ for some (collection of) value(s) of b.[1] Level sets are also called contour lines or contour curves.

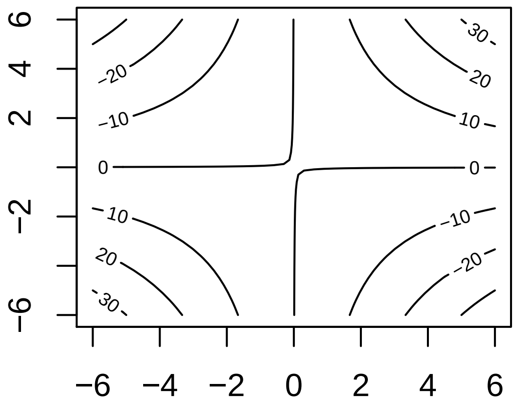
Some examples are:

```r
x_1 <- x_2 <- seq(-6, 6, 0.3)

func_1 <- function(x_1, x_2){x_1*x_2}
z_1 <- outer(x_1, x_2, func_1)
persp(x_1, x_2, z_1, theta = 30, phi = 0)
```



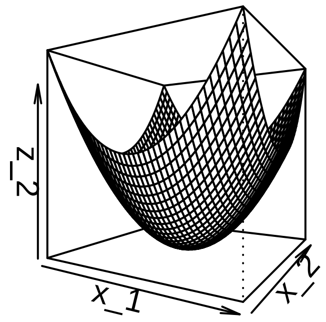
```r
contour(x_1, x_2, z_1)
```



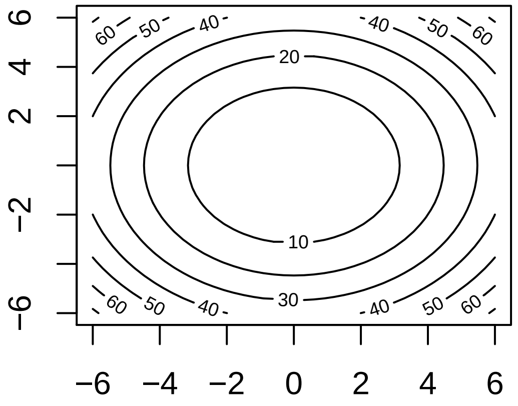
```r
func_2 <- function(x_1, x_2){x_1^2+x_2^2}
z_2 <- outer(x_1, x_2, func_2)
```

[1] Alternatively, the set $x \in \{f^{-1}(b)\}$.

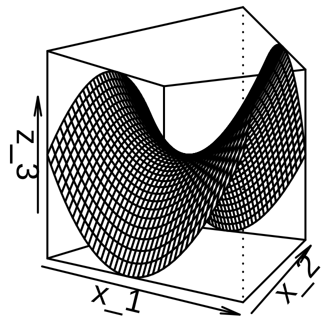
```r
persp(x_1, x_2, z_2, theta = 30, phi = 0)
```



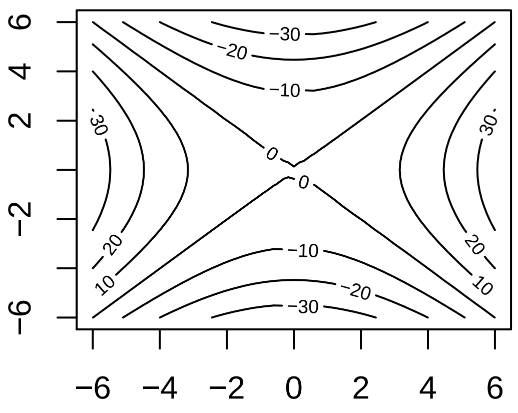
```r
contour(x_1, x_2, z_2)
```



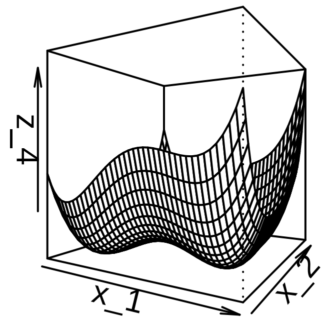
```r
func_3 <- function(x_1, x_2){x_1^2-x_2^2}
z_3 <- outer(x_1, x_2, func_3)
persp(x_1, x_2, z_3, theta = 30, phi = 0)
```



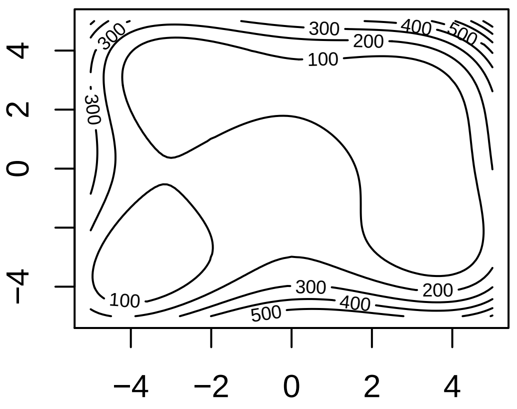
```
contour(x_1, x_2, z_3)
```



```
func_4 <- function(x_1, x_2){(x_1^2+x_2-11)^2 +
    (x_1+x_2^2-7)^2} #Himmelblau's function
z_4 <- outer(x_1, x_2, func_4)
persp(x_1, x_2, z_4, theta = 30, phi = 0)
```



```
y_1 <- y_2 <- seq(-5, 5, 0.1)
z_5 <- outer(y_1, y_2, func_4)
contour(y_1, y_2, z_5)
```



In which direction does the value of the function not change? In order to keep the value of the function at, say, $f(x) = 10$ we need to follow the curve given by the level set $f^{-1}(10)$. The direction which we need to follow at point x is the tangent direction.

Additionally, the direction in which the function does not change must satisfy $D_v f = 0 \Rightarrow (\nabla f)^\top v = 0$, which implies that the tangent direction is orthogonal to the gradient.

Gradients can be computed using the package `pracma` and the function `pracma::grad`.

Gradients and Lagrangians

An alternate way to derive the Lagrangian from gradients is see via the following:

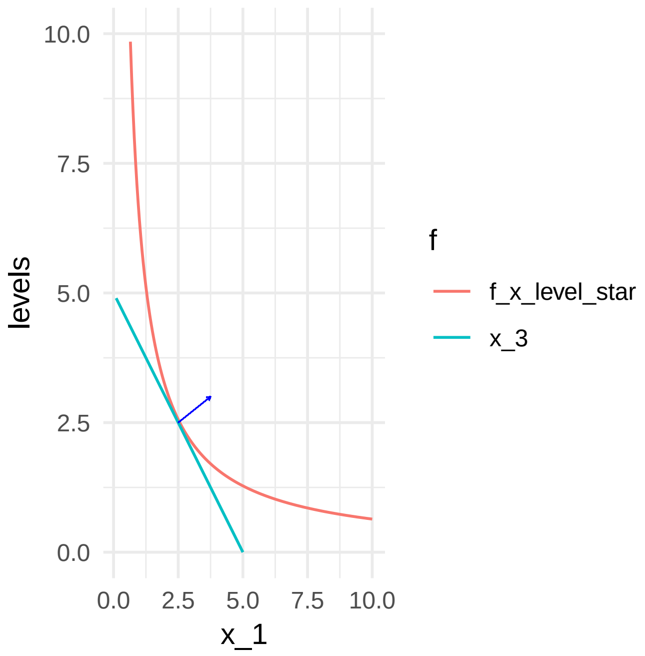
```
x_1 <- seq(0.1, 10, 0.05)
x_2 <- seq(0.1, 10, 0.05)
x_3 <- 5 - x_2

f_x_level_star <- 6.4/x_2 #set manually

data_plot_grad <- cbind(x_1,
                        f_x_level_star,
                        x_3
                        ) %>%
```

```r
  dplyr::as_tibble() %>%
  tidyr::gather(.,
                c(f_x_level_star, x_3),
                key = 'f',
                value = 'levels'
                )

ggplot(data_plot_grad, aes(x_1, levels, color = f)) +
  geom_line() +
  scale_y_continuous(limits = c(0, 10)) +
  geom_segment(aes(x = 2.5,
                   y = 2.5,
                   xend = 3.75, #set manually
                   yend = 3
                   ),
               color = "blue",
               arrow = arrow(length = unit(0.015, "npc")),
               size = 0.2
               ) +
  theme_minimal()
```



The gradient of f and h are respectively $[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}]$ and $[\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}]$. These point to the directions of maximum change and are orthogonal to the level sets of f, h.

Since the level sets and the constraints are tangent at the optimal implies that their respective gradients—orthogonal to the tangent—must lie on the same line.

$$[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}] = \lambda[\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_1}]$$

This yields exactly the Lagrangian function for the optimization program.

Several Equality Constraints

Consider the following variation on the maximization problem where there are several equality constraints now:

$$\max f(x) : x \in \mathbb{R}^n, x \geq 0$$

$$C_h = \{h_1(x) = c_1, \ldots, h_m(x) = c_m\}$$

The generalization from one to many equality constraints is straightforward.

Constraint Qualification

In the case of one constraint the optimal is not a critical point for the equality constraints, i.e., the qualification is:

$$[\frac{\partial h}{\partial x_1}(x^*), \ldots, \frac{\partial h}{\partial x_n}(x^*)] \neq (0, \ldots, 0)$$

Likewise, in the case of m equality constraints: $\{h_1(x) = c_1, \ldots, h_m(x) = c_m\}$, their *Jacobian* (first derivative matrix) must be *invertible* at the critical point x^*.

$$Dh(x^*) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1}(x^*), \ldots, \frac{\partial h_1}{\partial x_n}(x^*) \\ \frac{\partial h_2}{\partial x_1}(x^*), \ldots, \frac{\partial h_2}{\partial x_n}(x^*) \\ \vdots \\ \frac{\partial h_m}{\partial x_1}(x^*), \ldots, \frac{\partial h_m}{\partial x_n}(x^*) \end{bmatrix}$$

For the constraint Jacobian at the critical point to be invertible implies that the matrix above must have *full rank*, further equivalent to the condition that the determinant be non-zero at the critical point. More formally, it is said that $(h_1, \ldots, h_m)$ satisfy the *non-degenrate constraint qualification* (NDCQ) at x^* if matrix $Dh(x^*)$ is invertible at x^* (has full rank).

The Geometry of the Lagrangian

When there are m constraints: $h_1(x) = c_1, \ldots, h_m(x) = c_m$, the gradient of the objective function $\nabla f(x^*)$ at the optimal must be a linear combination of the gradients of the constraints $\sum_{i=1}^{m} \lambda_i \nabla h_i(x^*)$.

This is so because ∇h_i gives the directions of maximum increase of h_i. If the constraints $h_i = c_i$ are to be satisfied, we must move in the direction where h_i are constant (and equal to c_i) and hence in directions *orthogonal* (perpendicular) to ∇h_i and hence in directions *orthogonal* to $\sum_{i=1}^{m} \lambda_i \nabla h_i(x^*)$. In the same way, the contour lines for f are orthogonal to ∇f.

At the optimal, the objective function f must touch (be tangent to) the binding constraints and hence the gradient of f must lie in the *span* of the constraints' gradients: $\nabla f(x^*) = \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*)$.

Theorem: Given $f, h_1, \ldots, h_m : \mathbb{R}^n \to \mathbb{R} \in C^1$, where f is the objective function and $h_i = c_i$ are equality constraints which x must satisfy; and that h_i satisfiy the NDCQ condition; then there are $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$ such that (x^*, λ^*) is the critical point of the Lagrangian $\mathcal{L}(x, \lambda)$.

Hence

$$\mathcal{L}(x, \lambda) := f(x) - \lambda_1 \cdot (h_1(x) - c_1) + \ldots + \lambda_m \cdot (h_m(x) - c_m)$$

and

$$\frac{\partial \mathcal{L}}{\partial x_1}(x^*, \lambda^*) = 0, \ldots, \frac{\partial \mathcal{L}}{\partial x_n}(x^*, \lambda^*) = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_1}(x^*, \lambda^*) = 0, \ldots, \frac{\partial \mathcal{L}}{\partial \lambda_m}(x^*, \lambda^*) = 0$$

In total we get $n + m$ equations with $n + m$ unknowns: $(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m)$.

Inequality Constraints

Consider now a maximization program in $\mathbb{R}^2$ with one inequality constraint:

$$\max f(x_1, x_2), (x_1, x_2) \geq 0, g(x_1, x_2) \leq b$$

Binding Constraint

To make the discussion more concrete, reconsider the maximization of the objective function $f = x_1 x_2$ with an inequality constraint $g(x) = x_1^2 + x_2^2 \leq 1$.

We can see that for such a maximization program, the optimal must occur at the boundary of the constraint set $g(x) = b$. The level sets of f, g are tangent to each other, or equivalently their gradients lie on the same line:

$$\nabla f(x^*) = \lambda \nabla g(x^*)$$

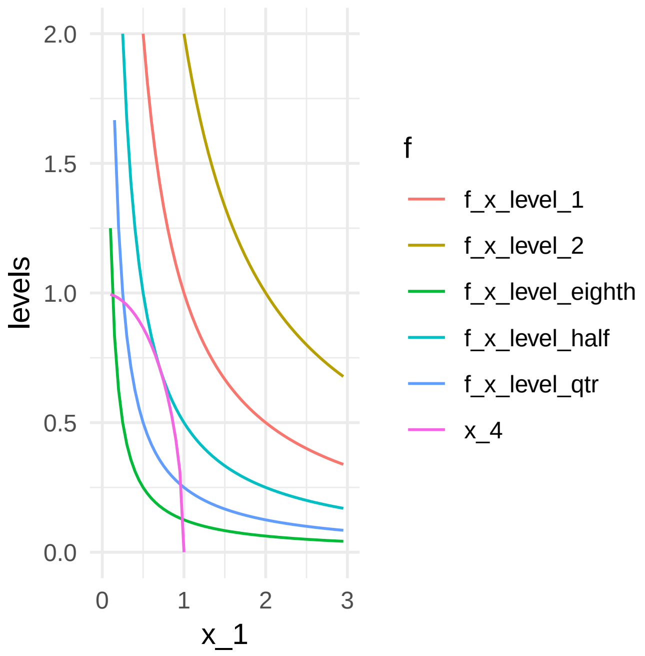
Additionally, however we note that $\lambda > 0$ i.e., the gradients point in the same direction and *not* in the opposite direction as is seen below.

```r
x_4 <- sqrt(1 - x_2^2) #inequality constraint

f_x_level_1 <- 1/x_2
f_x_level_half <- 1/(2*x_2)
f_x_level_qtr <- 1/(4*x_2)
f_x_level_eighth <- 1/(8*x_2)

data_plot_ineq <- cbind(x_1,
                        f_x_level_half,
                        f_x_level_1,
                        f_x_level_2,
                        f_x_level_eighth,
                        f_x_level_qtr,
                        x_4
                        ) %>%
  dplyr::as_tibble() %>%
  tidyr::gather(.,
                f_x_level_half:x_4,
                key = 'f',
                value = 'levels'
                )

ggplot(data_plot_ineq, aes(x_1, levels, color = f)) +
  geom_line() +
  scale_y_continuous(limits = c(0, 2)) +
  scale_x_continuous(limits = c(0, 3)) +
  theme_minimal()
```



As before, the constraint qualification condition holds—the optimal should not be a critical point of the inequality constraint.

Non-binding Constraint

When constraints are of the form $g(x) \leq b$, the optimal may as well lie strictly in the interior depending upon the objective function contour lines. In such cases, we can still form the Lagrangian provided we enforce the condition that $\lambda = 0$, which when combined with the slackness of the inequality: $g(x) - b < 0$ gives complementary slackness conditions: $\lambda \cdot (g(x) - b) = 0$.

Since we do not know which constraint is non-binding at the optimal, we cannot set $\frac{\partial \mathcal{L}}{\partial \lambda} = 0$ (which is equivalent to $g(x) - b = 0$). Hence such a condition is replaced by the complementary slackness condition, which implies that either the constraint is binding, in which case $\lambda > 0$ or the constraint is slack in which case $\lambda = 0$.

Theorem: Given functions $f, g : \mathbb{R}^2 \to \mathbb{R} \in C^1$ and that $g(x_1, x_2) \leq b$ (additionally, if $g(x^*) = b$, then $\nabla g(x^*) \neq 0$); then for the Lagrangian function there is (λ^*) such that the following are true:

1. $\nabla_{x^*} \mathcal{L} = 0$ (Primal stationary)

2. $\lambda^* \cdot (g(x^*) - b) = 0$ (Complementary slackness)
3. $\lambda^* \geq 0$ (Dual Feasibility)
4. $g(x^*) \leq b$ (Primal Feasibility)

Remarks

1. $\nabla_x \mathcal{L} = 0$ for both equality and inequality constraints.
2. $\nabla_\lambda \mathcal{L} = 0$ is true *only* for equality constraints.
3. Constraint qualification needs to be checked *only* for binding inequality constraints.
4. $\lambda = 0$ *only* for inequality constraints.
5. Complementary slackness must hold for inequality constraints.

Several Inequality Constraints

The generalization from one inequality constraint to several is fairly straightforward:

Theorem: Given functions $f, g_1, \ldots, g_k : \mathbb{R}^n \to \mathbb{R} \in C^1$ and $x^* \in \mathbb{R}^n$ is a local maximizer on the constraint set

$$C_g = \{g_1(x) \leq b_1, \ldots, g_k(x) \leq b_k\}$$

Assume without loss of generality that the first k_0 constraints are binding at x^* and the rest $k - k_0$ are slack. Also assume the nondegeneracy constraint qualification is satisfied by stipulating that the following constraint derivative matrix is invertible at the optimal x^*

$$Dg(x^*) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1}(x^*), \ldots, \frac{\partial g_1}{\partial x_n}(x^*) \\ \vdots \\ \frac{\partial g_{k_0}}{\partial x_1}(x^*), \ldots, \frac{\partial g_{k_0}}{\partial x_n}(x^*) \end{bmatrix}$$

That is, the constraint Jacobian at the critical point is invertible and has full rank.

Then the Lagrangian

$$\mathcal{L}(x, \lambda) := f(x) - \lambda_1 \cdot (g_1(x) - b_1) - \ldots - \lambda_k(g_k(x) - b_k)$$

has the critical points (x^*, λ^*) such that:

1. $\nabla_x \mathcal{L}(x^*) = 0$ (Primal stationary)
2. $\lambda_1^* \cdot (g_1(x) - b_1) = 0, \ldots, \lambda_k^* \cdot (g_k(x) - b_k) = 0$ (Complementary slackness)
3. $\lambda_1^* \geq 0, \ldots, \lambda_k^* \geq 0$ (Dual feasible)
4. $g_1(x^*) \leq b_1, \ldots, g_k(x^*) \leq b_k$ (Primal feasible)

The General Case: Mixed Constraints

The Karush-Kuhn-Tucker (KKT) conditions are first order necessary conditions for a general optimization program including equality as well as inequality constraints. If there are no inequality constraints, the KKT conditions are equivalent to the method of Lagrange multipliers.

We consider the general case:

$$\max f(x), x \in \mathbb{R}^n, x \geq 0 :$$

$$C_g = \{g_1(x) \leq b_1, \ldots, g_k(x) \leq b_k\}$$

$$C_h = \{h_1(x) = c_1, \ldots, h_m(x) = c_m\}$$

We assume $f, \{g_i\}_{i=1}^k, \{h_j\}_{j=1}^m : \mathbb{R}^n \to \mathbb{R} \in C^1$.

For the inequality constraint set C_g, we assume without loss of generality that the first k_0 constraints are binding at x^*. We assume that the following derivative matrix is invertible at the optimal (NDCQ) (has full rank):

$$\begin{bmatrix} \frac{\partial g_1}{\partial x_1}(x^*), \ldots, \frac{\partial g_1}{\partial x_n}(x^*) \\ \vdots \\ \frac{\partial g_{k_0}}{\partial x_1}(x^*), \ldots, \frac{\partial g_{k_0}}{\partial x_n}(x^*) \\ \frac{\partial h_1}{\partial x_1}(x^*), \ldots, \frac{\partial h_1}{\partial x_n}(x^*) \\ \vdots \\ \frac{\partial h_m}{\partial x_1}(x^*), \ldots, \frac{\partial h_m}{\partial x_n}(x^*) \end{bmatrix}$$

We form the Lagrangian as before:

$$\mathcal{L}(x, \lambda, \mu) := f(x) - \sum_{i=1}^{k} \lambda_i \cdot (g_i(x) - b_i) - \sum_{j=1}^{m} \mu_j \cdot (h_i(x) - c_i)$$

The critical points of the Lagrangian are found from the FOC:

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) := \nabla f(x^*) - \sum_{i=1}^{k} \lambda_i \nabla g_i(x^*) - \sum_{j=1}^{m} \mu_j \nabla h_j(x^*) = 0$$

The Lagrangian above has multipliers $\lambda^* \in \mathbb{R}^k, \mu^* \in \mathbb{R}^m$ such that

1. $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$ (Primal stationarity)
2. $\lambda_1 \cdot (g_1(x^*) - b_1) = 0, \ldots, \lambda_k \cdot (g_k(x^*) - b_k) = 0$ (Complementary slackness)
3. $h_1(x^*) - c_1 = 0, \ldots, h_m(x^*) - c_m = 0$ (Primal feasibility)
4. $\lambda_1^* \geq 0, \ldots, \lambda_k^* \geq 0$ (Dual feasibility)
5. $g_1(x^*) - b_1 \leq 0, \ldots, g_k(x^*) - b_k \leq 0$ (Primal feasibility)

Interpreting the Lagrange Multiplier

In plain language, the Lagrange multiplier measures the sensitivity of the optimal value of the objective function to changes in the constraints' right hand sides.

We illustrate this idea using a simple maximization program in two dimensions and with one equality constraint:

$$\max f(x), x \in \mathbb{R}^2, x \geq 0 :$$
$$h(x) = c$$

The central idea is to consider c as a parameter that varies from problem to problem. Given c, $(x_1^*(c), x_2^*(c))$ is the optimal as a function of the constraint parameter c; $f(x_1^*(c), x_2^*(c))$ is the optimal value as a function of constraint parameter; and $\lambda^*(c)$ is the corresponding multiplier.

Theorem: Given $f, h : \mathbb{R}^2 \to \mathbb{R} \in C^1$; given $x_1^*(c), x_2^*(c), \lambda^*(c) \in C^1$ and NDCQ condition:

$$\lambda^*(c) = \frac{d}{dc} \cdot [f(x_1^*(c), x_2^*(c))]$$

The same idea can be generalized for the case with several mixed constraints.

Application: Portfolio Optimization

The original mean-variance framework admits a quadratic program that may be solved via `quadprog()`. However, as has been argued effectively, variance is but a poor measure of risk. As opposed to variance, which penalizes both positive and negative deviations from the mean equally, a true risk measure should penalize only negative deviations since risk emanates from losses but *not* from gains.

Hence a general portfolio optimization program where risk is measured via some function $\rho(\cdot)$ is

$$\begin{gathered}\min \rho(w) : \\ w^\top e = 1 \\ w^\top \mu \geq \mu_* \\ w \geq 0\end{gathered}$$

Some common measures of portfolio risk are “variance”, “Value-at-Risk” (VaR); “Average Value-at-Risk” (AVaR); and many others. For the case when the risk measure is variance, the resulting optimization framework is the mean-variance framework.

Constraints

Depending upon institutional frameworks, there are many extra constraints that a portfolio manager may need to satisfy. We discuss some prominent ones below that are either linear or quadratic in nature.

Long-only Constraint

We’ve been imposing this constraint implicitly whenever we’ve insisted on the constraint $w \geq 0$. $w_j < 0$ implies that for security j one has negative weight—implying the use of borrowed money for use of asset j.

Holding Constraints

For constraints that stipulate that the securitiy weights be between maximal and minimal limits:

$$L_i \leq w_i \leq U_i$$

Turnover Constraints

High turnovers may lead to high transaction costs. There could be mandates for turnover constraints on individual securities as well as on the entire portfolio. If the current weights are w_0 and targeted weights are w then the turnover constraint for security i manifests as:

$$|w - w_0|_i \leq U_i$$

And for the whole portfolio: $\sum_{i=1}^{n} |w - w_0|_i \leq U_{port}$

Tracking Error Constraints

A very coarse classification of fund management is between "active" and "passive" funds. Active managers assume that there are pockets of inefficiency in financial markets and try to exploit them to beat the market. Passive fund managers assume that markets are more or less efficient and aim to mimic the market index as a whole. Today passive indexing is an extremely popular type of fund management.

Often managers of passive and/or index funds are tasked with *following* an index, say the S&P500. The idea is that the fund manager replicates the index performance—if the index posts high gains, the investors in the fund reap benefits; and if the index does poorly, so do the investors.

If w_b are the benchmark market capitalization weights for an index and r the return vector, the benchmark returns are $w_b^\top r = r_b$. A passive fund manager may choose to impose the following restriction on portfolio weights:

$$\|w - w_b\| \leq M$$

The most common metric to be used for this purpose is the “tracking error” which is the variance of the difference between the portfolio and benchmark returns.

$$\mathrm{TE}_p = \mathrm{var}(r_p - r_b) = \mathrm{var}(w^\top r - w_b^\top r)$$
$$\mathrm{TE}_p = (w - w_b)^\top \mathrm{var}(r)(w - w_b) = (w - w_b)^\top \Sigma (w - w_b)$$

To insist that the tracking error be less than a critical value introduces the following constraint:

$$(w - w_b)^\top \Sigma (w - w_b) \leq \sigma_{TE}^2$$

Hence the benchmark tracking problem can be framed as:

$$\min_w \, (w - w_b)^\top \Sigma (w - w_b) :$$
$$w^\top e = 1$$

Computation: Portfolio Optimization

Let’s revisit the mean-variance problem from before.

There are two normally distributed common stocks whose expected returns are $\mu^\top = (1.8\%, 2.5\%)$ with covariance matrix Σ:

$$\Sigma = \begin{bmatrix} 1.68, 0.34 \\ 0.34, 3.09 \end{bmatrix}$$

The minimization program was:

$$\min 1.68 w_1^2 + 3.09 w_2^2 + 2 * 0.34 w_1 w_2 :$$
$$w_1 + w_2 = 1$$
$$0.018 w_1 + 0.025 w_2 \geq 0.018$$
$$w_1, w_2 \geq 0$$

However, in the new problem suppose there is a holding constraint: the manager cannot have any stock weigh more than 60%.[2]

Hence the new minimization program becomes:

$$\min 1.68w_1^2 + 3.09w_2^2 + 2 * 0.34w_1w_2 :$$

$$w_1 + w_2 = 1$$

$$0.018w_1 + 0.025w_2 \geq 0.018$$

$$w_1 \leq 0.60$$

$$w_2 \leq 0.60$$

$$w_1, w_2 \geq 0$$

We can solve this via `quadprog` using the `solve.QP()` function.

```r
D_mat <- matrix(c(1.68, 0.34, 0.34, 3.09), 2, 2)
d_vec <- matrix(0, 0, nrow = 2, ncol = 1)
A_mat <- matrix(c(1, 1, 0.018, 0.025, -1, 0, 0, -1),
                nrow = 4,
                byrow = T
                )
b_vec <- c(1, 0.018, -0.60, -0.60)
m_eq <- 1

quadprog::solve.QP(2*D_mat, d_vec, t(A_mat), b_vec, m_eq) #note
```

```
## $solution
## [1] 0.6 0.4
##
## $value
## [1] 1.2624
```

[2]This constraint could be needed to avoid over-reliance on one stock lest it perform poorly sometime and take the portfolio down with it. Note that the previous problem where this constraint does not feature suggests best weight for the first stock to be 0.67, which will be infeasible in this case.

```
##
## $unconstrained.solution
## [1] 0 0
##
## $iterations
## [1] 3 0
##
## $Lagrangian
## [1] 2.880 0.000 0.592 0.000
##
## $iact
## [1] 1 3
```

References

Bloomfield, Victor A. 2014. *Using R for Numerical Analysis in Science and Engineering.* CRC Press.

Jones, Owen, Robert Maillardet, and Andrew Robinson. 2014. *Introduction to Scientific Programming and Simulation Using R.* Second Edition. CRC Press.

Rachev, Svetlozar T., Stoyan V. Stoyanov, and Frank J. Fabozzi. 2008. *Advanced Stochastic Models, Risk Assessment, and Portfolio Optimization.* Hoboken, New Jersey: John Wiley; Sons.

Simon, Carl, and Lawrence Blume. 1994. *Mathematics for Economists.* W W Norton.