Constrained optimization: equality constraints

Lluís Garrido – lluis.garrido@ub.edu

November 2016

Abstract

This laboratory is focused on constrained optimization and, in particular, con equality constraints. The constrained optimization problem will be easily transformed into an unconstrained optimization problem using Lagrange multipliers.

1 Equality constraints: KKT conditions

Let us begin with a summary of equality constrained optimization[1].

Consider the problem of minimizing $f(x)$ subject to the constraints $h_i(x) = 0$ for $i = 1 \ldots m$, where $x \in R^n$. Without any constraint, $m = 0$, the necessary condition for optimality is $\nabla f(x) = 0$.

Let us now examine the case where $m = 1$, that is, a single constraint. With the constraint $h(x) = 0$, we require that x lies on the graph of the (nonlinear) equation $h(x)$, see figure 1. Assume that x^* is the optimal point we are looking for. The steepest descent direction at x^*, $-\nabla f(x^*)$, is orthogonal to the tangent of the contours of f through the point x^*. The same reasoning tells us that $\nabla h(x^*)$ also is orthogonal to the curve $h(x) = 0$ at x^*. Otherwise, x^* wouldn't be the optimum, see figure 1.

In addition, $\nabla f(x^*)$ must be orthogonal to the tangent of the curve $h(x) = 0$ at x^*. This can be easily demonstrated and here we just give an insight. Assume that $c(t)$ is a curve $\{c = c(t) : t_0 \leq t \leq t_1\}$ such that $h(c(t)) = 0$. In other words, $c(t)$ is the feasible curve with respect to the constraint $h(x) = 0$. For t^*, we have that $c(t^*) = x^*$, the optimal value. Observe that $f(c(t))$ has a minimum at $t = t^*$, that is, the derivative of $f(c(t))$ with respect to t must vanish (i.e. be zero) at $t = t^*$. The derivative of $f(c(t))$ with respect to t at $t = t^*$ is

$$\left.\frac{d}{dt} f(c(t))\right|_{t=t^*} = c'(t)^T \left.\nabla f(c(t))\right|_{t=t^*} = c'(t^*)^T \nabla f(x^*)$$

The previous expression indicates us that $\nabla f(x^*)$ must be orthogonal to the tangent of the curve $h(x) = 0$ at x^*.

[1] The text written in this section has been obtained from Griva, I.; Nash, S.; Sofer, A., "Linear and nonlinear optimization", SIAM and http://www.pitt.edu/~jrclass/opt/notes3.pdf.

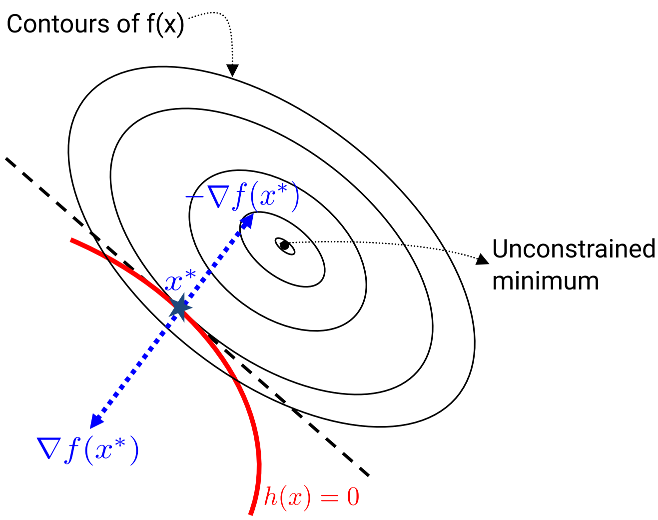


Figure 1: For a single constraint, $m = 1$, we seek a point x^*such that $-\nabla f(x^*) = \lambda \nabla h(x^*)$.

Therefore, $-\nabla f(x^*)$ and $\nabla h(x^*)$ must both lie along the same line. That is, for some $\lambda \in R^+$ we must have

$$-\nabla f(x^*) = \lambda \nabla h(x^*)$$

Thus, if x^* is a minimizer, the necessary condition reduces to

$$\nabla_{x,\lambda} f(x^*) + \lambda \nabla_{x,\lambda} h(x^*) = 0$$

where $\nabla_{x,\lambda}$ referers to the gradient computation with respect x and λ. If not specified, the gradient is computed only with respect to x.

For the general case, in which we have constraints $g_i(x) = 0$ for $j = 1 \dots m$, the above necessary condition should hold for each constraint. Assuming that $\nabla g_i(x^*)$ are linearly independent (or equivalenly, the Jacobian matrix has full row rank), the point $x^* \in R^n$ must satisfy the following necessary condition for some $\lambda^* \in R^m$, that is

$$\nabla_{x,\lambda} f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*) = 0 \qquad h_i(x^*) = 0 \quad \forall i$$

The latter conditions are known as the **Karush-Kuhn-Tucker** (KKT) conditions.

Example 1

Assume we want to minimize

$$f(x) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1 x_2 - 3x_2 \quad \text{such that} \quad x_1 + x_2 = 3$$

Observe that $m = 1$ (only one equality constraint) and that our constraint is $h(x) = x_1 + x_2 - 3 = 0$. The Lagrangian is

$$L(x, \lambda) = \left(\frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1 x_2 - 3\right) + \lambda(x_1 + x_2 - 3)$$

where $x \in R^3$ and $\lambda \in R$. The KKT conditions yield

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= x_1 - x_2 + \lambda = 0 \\
\frac{\partial L}{\partial x_2} &= x_2 - x_1 - 3 + \lambda = 0 \\
\frac{\partial L}{\partial \lambda} &= x_1 + x_2 - 3 = 0
\end{aligned}$$

The minimizer can be found using the gradient descent. In this case, however, we have a set of linear equations and thus they can be easily solved. The minimizer is $x_1^* = 0.75$, $x_2^* = 2.25$, $\lambda^* = 1.5$.

Example 2

We want to find the area of the largest rectangle that can be inscribed inside the ellipse

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1$$

That is, we want to *maximize* $f(x_1, x_2) = 4x_1x_2$ with $h(x_1, x_2) = x_1^2/a^2 + x_2^2/b^2 - 1 = 0$. The Lagrangian is

$$L(x, \lambda) = -4x_1x_2 + \lambda\left(\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1\right)$$

The previous problem can analytically solved

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= -4x_2 + 2\lambda\frac{x_1}{a^2} = 0 \\
\frac{\partial L}{\partial x_2} &= -4x_1 + 2\lambda\frac{x_1}{b^2} = 0 \\
\frac{\partial L}{\partial \lambda} &= \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - 1 = 0
\end{aligned}$$

The solution to the problem is

$$x_1^2 = \frac{a^2}{2} \qquad x_2^2 = \frac{b^2}{2} \qquad \lambda = 2ab$$

Observe that the solution is unique (assuming $x_1 > 0$ and $x_2 > 0$).

Exercise

You are requested to numerically find the solution to exercise 2 using the gradient descent. The main difficulty that arises when solving these types of problems is the fact that the condition $\nabla_{x,\lambda} L(x, \lambda) = 0$ corresponds generally to a **saddle point**, not a minimum (nor a maximum). This is the case for the problem of exercise 2.

There are several numerical techniques that allow to find a point x^* that corresponds to $\nabla_{x,\lambda} L(x^*, \lambda^*) = 0$, the saddle point. One way to tacke the problem is to "construct a new function, related to the Lagrangian, that (ideally) has a minimum at (x^*, λ^*). This new function can be considered as 'distorting' the Lagrangian at infeasible points so as to create a minimum at (x^*, λ^*). Unconstrained minimization techniques can then be applied to the new function. This approach can make it easier to guarantee convergence to a local solution, but there is the danger that the local convergence properties of the method can be damaged. The 'distortion' of the Lagrangian function can lead to a 'distortion' in the Newton equations for the method. Hence the behavior of the method near the solution may be poor unless care is taken."

Another way to tackle the condition$\nabla_{x,\lambda} L(x, \lambda) = 0$ is to maintain feasibility at every iteration. That is, to ensure that the updates x^k follow the implicit curve $h(x) = 0$. For the problem we are considering here it is relatively easy. Assume we start from a point x^0 that satisfies $h(x^0) = 0$, that is it satisfies the constraint. The algorithm can be summarized as follows:

1. Compute the gradient $\nabla L(x^k)$ (observe that we compute the gradient of the Lagrangian with respect to x).

2. Compute an estimate of λ by computing the value of λ that minimizes $\left\|\nabla L(x^k)\right\|^2$.

3. Assume that the update is $x^{k+1} = x^k - \alpha^k \nabla L(x^k)$. For each candidate update x^{k+1}, project it over the constraint $h(x) = 0$. Find the α^k value that decreases the $L(x^{k+1})$ with respect to $\nabla L(x^k)$.

4. Goto step 1 and repeat until convergence.

For the problem we are considering observe that

$$\nabla L(x, \lambda) = \left(-4x_2 + 2\frac{x_1}{a^2}, -4x_1 + 2\frac{x_2}{b^2}\right)^T$$

For a given iteration x^k, the estimate λ can be computed by minimizing $\left\|\nabla L(x^k)\right\|^2$. Thus, the optimum value λ is:

$$\frac{d}{d\lambda} \left\|\nabla L(x)\right\|^2 = 0 \quad \Rightarrow \quad \lambda = \frac{2a^2b^2(a^2 + b^2)x_1x_2}{a^4x_2^2 + b^4x_1^2}$$

The Python code is available with this lab.

2 An exercise: PCA analysis

Let us know focus on the Principal Component Analysis problem, that is,

$$\max_w w^T A w \ \text{ subject to } \ w^T w = 1$$

where A is the covariance matrix of the considered data.

There is a closed solution to the problem. For that issue consider the equivalent unconstrained problem

$$\max_w \frac{w^T A w}{w^T w}$$

An analytical solution to this problem can be found using $\nabla_w = 0$. Thus,

$$\left(Aw + (w^T A)^T\right)(w^T w)^{-1} - 2w^T A w w (w^T w)^{-2} = 0$$

$$Aw - w^T A w w (w^T w)^{-1} = 0$$

That is,

$$\frac{Aw}{w^T A w} = \frac{w}{w^T w}$$

The solution of the former problem corresponds to the eigenvector of maximum eigenvalue. Figure 2 shows the Pythons's code that allows to perform such eigenvector analysis.

Exercise

You are know asked to use the KKT conditions to see if you are able to obtain the same result as the one obtained with the closed solution. You may use the projected gradient descent to obtain the solution.

Report

You are asked to deliver an *individual* report of the exercise proposed in section 2. Just explain each of the steps you have followed. If you want to include some parts of code, please include it within the report. Do not include it as separate files. You may just deliver the Python notebook if you want.

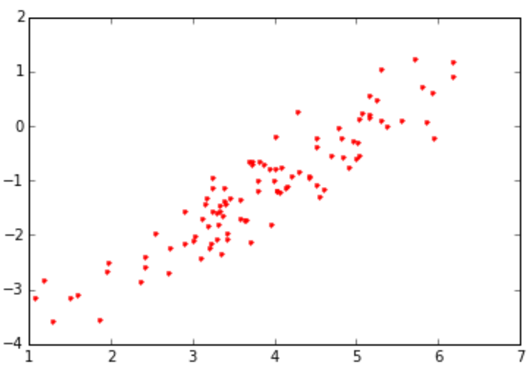
The deadline to deliver this report is December 13th at 3 p.m. (15h).

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

        %matplotlib inline

        m1 = [4.,-1.]
        s1 = [[1,0.9],[0.9,1]]
        c1 = np.random.multivariate_normal(m1,s1,100)
        plt.plot(c1[:,0],c1[:,1],'r.')
```

```
Out[1]: [<matplotlib.lines.Line2D at 0x7fd3ba98d5d0>]
```




```
In [2]: vaps,veps = np.linalg.eig(np.cov(c1.T))
        idx = np.argmax(vaps)

        plt.plot(c1[:,0],c1[:,1],'r.')
        plt.arrow(np.mean(c1[:,0]),np.mean(c1[:,1]),
                  vaps[idx]*veps[0,idx],vaps[idx]*veps[1,idx],0.5,
                  linewidth=1,head_width=0.1,color='blue')
```

```
Out[2]: <matplotlib.patches.FancyArrow at 0x7fd3baa21950>
```

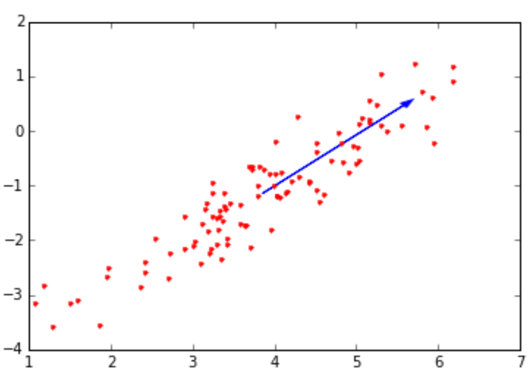


Figure 2: Pythons' code that allows to perform PCA analysis of a set of data. The red dots of the figure show the considered data, the blue arrow shows the eigenvector of maximum eigenvalue. Code, available with this lab, thanks to Oriols' notebook.