Penalty Functions and Constrained Optimization

Kurt Bryan and Yosi Shibberu

Introduction

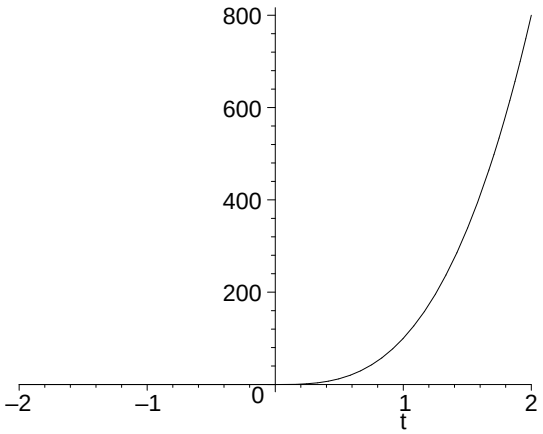
We've had a pretty thorough introduction to unconstrained optimization. Now we'll briefly consider constrained optimization. This is a rather more difficult subject. It makes use of more heavy duty linear algebra and geometry, and the results and algorithms are generally more complicated and technical. As such, we'll really just look at one basic technique, that has the advantage of turning constrained problems into unconstrained problems! We can then attack the problem using any of the unconstrained methods we've developed.

For motivation, consider minimizing the function $f(x) = x^4$ subject to the constraint $x \geq 1$ (and ignore the fact that this is easy). The algorithms we've seen so far are strictly for unconstrained optimization problems, and in this case that won't suffice; the constraint that $x \geq 1$ clearly influences the location of the minimum. The method of *penalty functions* provides a way to attack constrained optimization problems using algorithms for unconstrained problems.

Here's how it works, via the example above. Define a function $\phi(t)$ by

$$\phi(t) = \begin{cases} 0 & \text{for } t < 0 \\ kt^3 & \text{for } t \geq 0 \end{cases} \tag{1}$$

where k is some positive constant. The function ϕ is a *penalty* function. It "penalizes" any number t which is greater than zero (from the point of view of minimization). Here's a plot of ϕ for $k = 100$:

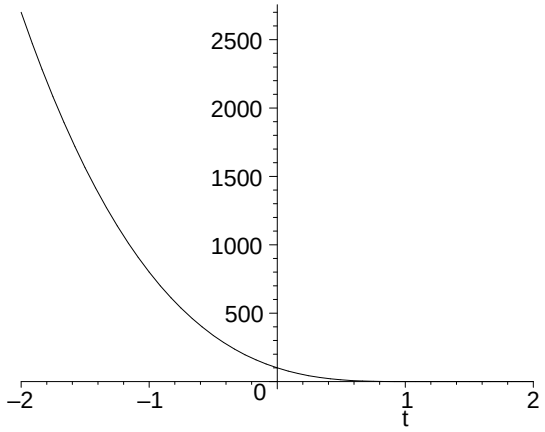


The penalty function is also twice differentiable, even through zero.

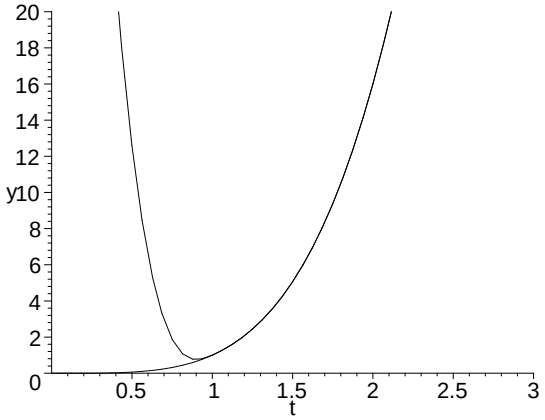
The penalty function let's us attack the constrained problem by turning it into an unconstrained problem, as follows. The constraint $x \geq 1$ is equivalent to $1 - x \leq 0$. Define a modified or "penalized" objective function

$$\tilde{f}(x) = f(x) + \phi(1 - x) = x^4 + \phi(1 - x).$$

The function $\tilde{f}(x)$ is identical to f if $1 - x \leq 0$, i.e., if $x \geq 1$, but rises sharply if $x < 1$. The additional $\phi(1 - x)$ term penalizes an optimization algorithm for choosing $x < 1$. Here's a plot of just the penalty term, $\phi(1 - x)$, with $k = 100$:



Here's a plot of $f(x)$ and $\tilde{f}(x)$, on a more compressed vertical scale:



We can approximately minimize $f(x)$ subject to the constraint $x \geq 1$ by running an unconstrained algorithm on the penalized objective function $\tilde{f}(x)$; the penalty term will strongly "encourage" the unconstrained algorithm to choose the best x which is greater than or equal to one. The penalty term is also twice differentiable, so it should not cause any trouble in an optimization algorithm which relies on first or second derivatives. The first

and second derivatives of $\phi(t)$ are just

$$\phi'(t) = \begin{cases} 0 & \text{for } t < 0 \\ 3kt^2 & \text{for } t \geq 0 \end{cases} \qquad \phi''(t) = \begin{cases} 0 & \text{for } t < 0 \\ 6kt & \text{for } t \geq 0 \end{cases}$$

If you run an unconstrained algorithm like golden section on $\tilde{f}(x)$ in this case (with $k = 100$) you find that the minimum is at $x = 0.9012$; the penalty approach didn't exactly solve the problem, but it is reasonably close.

In fact, a reasonable procedure would be to increase the constant k, say by a factor of 10, and then re-run the unconstrained algorithm on $\tilde{f}$ using 0.9012 as the initial guess. Increasing k enforces the constrained more rigorously, while using the previous final iterate as an initial guess speeds up convergence (since we expect the minimum for the larger value of k isn't that far from the minimum for the previous value of k). In this case increasing k to 10^4 moves the minimum to $x = 0.989$. We could then increase k and use $x = 0.989$ as an initial guess, and continue this process until we obtain a reasonable estimate of the minimizer.

The General Case:

In general we want to minimize a function $f(\mathbf{x})$ of n variables subject to both equality and inequality constraints of the form

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \tag{2}$$
$$h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, n. \tag{3}$$

You should convince yourself that any equality or inequality constraints can be cast into the above forms. The set of $\mathbf{x}$ in n dimensional space which satisfy the constraints is called the *feasible set*, although it may be empty if the constraints are mutually contradictory.

We will call $\phi(\lambda, t)$ for $\lambda \geq 0$, $t \in \mathbb{R}$, a *penalty* function if

1. ϕ is continuous.
2. $\phi(\lambda, t) \geq 0$ for all λ and t.
3. $\phi(\lambda, t) = 0$ for $t \leq 0$ and ϕ is strictly increasing for both $\lambda > 0$ and $t > 0$.

It's also desirable if ϕ has at least one continuous derivative in t, preferably two.

A typical example of a penalty function would be

$$\phi(\lambda, t) = \begin{cases} 0 & \text{for } t < 0 \\ \lambda t^n & \text{for } t \geq 0 \end{cases} \tag{4}$$

where $n \geq 1$. This function has $n - 1$ continuous derivatives in t, so taking $n = 3$ yields a C^2 penalty function.

To minimize $f(\mathbf{x})$ subject to constraints (2) and (3), define a modified objective function by

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} \phi(\alpha_i, g_i(\mathbf{x})) + \sum_{i=1}^{n} (\phi(\beta_i, h_i(\mathbf{x})) + \phi(\beta_i, -h_i(\mathbf{x})))$$

where the α_i and β_i are positive constants that control how strongly the constraints will be enforced. The penalty functions ϕ in the first sum modify the original objective function so that if any inequality constraint is violated, a large penalty is invoked; if all constraints are satisfied, no penalty. Similarly the second summation penalizes equality constraints which are not satisfied, by penalizing both $h_i(\mathbf{x}) < 0$ and $h_i(\mathbf{x}) > 0$. We minimize the function $\tilde{f}$ with no constraints, and count on the penalty terms to keep the solution near the feasible set, although no finite choice for the penalty parameters typically keeps the solution in the feasible set. After having minimized $\tilde{f}$ with an unconstrained method, for a given set of α_i and β_i, we may then increase the α_i and β_i and use the terminal iterate as the initial guess for a new minimization, and continue this process until we obtain a sufficiently accurate minimum.

Example: Let us minimize the function $f(x, y) = x^2 + y^2$ subject to the inequality constraint $x + 2y \geq 6$ and the equality constraint $x - y = 3$. In this case the constraints can by written as

$$\begin{aligned} g_1(x, y) &\leq 0, \\ h_1(x, y) &= 0, \end{aligned}$$

where $g_1(x, y) = 6 - x - 2y$ and $h_1(x, y) = 3 - x + y$. We use the penalty function defined in equation (4) with $\alpha_1 = 5$ and $\beta_1 = 5$ to start. The modified objective function is

$$\tilde{f}(x, y) = f(x, y) + \phi(5, g_1(x, y)) + \phi(5, h_1(x, y)) + \phi(5, -h_1(x, y)).$$

Run any standard unconstrained algorithm on this, e.g., a BFGS quasi-Newton method; the minimum occurs at $x = 3.506$ and $y = 1.001$. The equality constraint is violated ($3 - x + y = 0.494$), as is the inequality constraint ($6 - x - 2y = 0.449 > 0$). To increase the accuracy with which the constraints are enforced, increase the penalty parameters. It is very helpful to use the final estimate from the more modest penalty parameters as the initial guess for the larger parameters. With $\alpha_1 = \beta_1 = 50$ we obtain $x = 3.836$ and $y = 1.008$. Increasing $\alpha_1 = \beta_1 = 500$ we obtain $x = 3.947, y = 1.003$. The actual answer is $x = 4$, $y = 1$. Increasing the penalty parameters does improve the accuracy of the final answer, but it will also slow down the unconstrained algorithm's convergence, for $\tilde{f}(\mathbf{x})$ will then have a very large gradient and the algorithm will spend a lot of time hunting for an accurate minimum.

Under appropriate assumptions one can prove that as the penalty parameters are increased without bound, any convergent subsequence of solutions to the unconstrained penalized problems must converge to a solution of the original constrained problem.

Pros and Cons of Penalty Functions

The obvious advantage to the penalty function approach is that we obtain a "hands-off" method for converting constrained problems of any type into unconstrained problems. Also, we don't have to worry about finding an initial feasible point (sometimes a problem).

Another advantage to the penalty function approach is that (in my humble experience) many constraints in the real world are "soft", in the sense that they need not be satisfied precisely. The penalty function approach is well-suited to this type of problem.

The drawback to penalty function methods is that the solution to the unconstrained penalized problem will not be an exact solution to the original problem (except in the limit as described above). In some cases penalty methods can't be applied because the objective function is actually undefined outside the feasible set. I've worked on problems like this, where computing the objective function involves solving a PDE on some region and the independent variables control the geometry of the region. Infeasible values of the independent variables correspond to geometrically impossible shapes!

Another drawback to penalty methods is that as we increase the penalty parameters to more strictly enforce the constraints, the unconstrained formulation becomes very ill-conditioned, with large gradients and abrupt function changes. It also turns out that there are more efficient (but more elaborate and difficult) methods for approaching constrained optimization problems, but they are beyond what we'll cover in this course.

Barrier Function Methods

These are closely related to penalty function methods, and in fact might as well be considered a type of penalty function method. These methods are generally applicable only to inequality constrained optimization problems. Barrier methods have the advantage that they always maintain feasible iterates, unlike the penalty methods above.

The most common is the *log barrier* method. Suppose we have an objective function $f(\mathbf{x})$ with inequality constraints $g_i(\mathbf{x}) \leq 0$ for $1 \leq i \leq m$. Form a modified or penalized objective function

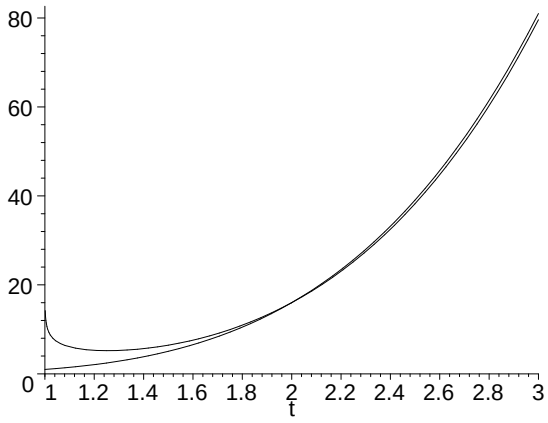
$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) - \sum_{i=1}^{m} r_i \ln(-g_i(\mathbf{x}))$$

where the $r_i > 0$. Notice that $\tilde{f}(\mathbf{x})$ is undefined if any $g_i(\mathbf{x}) \geq 0$, so we can only evaluate $\tilde{f}$ in the *interior* of the feasible region. However, even inside the feasible region the penalty term is non-zero (but it becomes an "anti-penalty" if $g_i \leq -1$).

Suppose we start some choice for the r_i and with initial feasible point $\mathbf{x}_0$, and minimize $\tilde{f}$. The terminal point $\mathbf{x}_k$, must be a feasible point, because the log terms in the definition of $\tilde{f}$ form a barrier of infinite height which prevents the optimization routine from leaving the interior of the feasible region.

Example: Here's a 1D example with objective function $f(t) = t^4$ and constraint $t \geq 1$.

The penalized objective function is $\tilde{f}(t) = t^4 - 2\ln(t-1)$ (so I took $r_1 = 2$):



In general a barrier method works in a similar way to the penalty methods above. We start with some positive r_i and feasible point $\mathbf{x}_0$. Minimize $\tilde{f}$ using an unconstrained algorithm. Now *decrease* the value of the r_i and re-optimize, using the final iterate as an initial guess for the newly decreased r_i. Continue until an acceptable minimum is found.

One point on which we need to be careful is the line search—you don't want to evaluate $\tilde{f}$ at any point outside the feasible set (or at least you need to deal with this gracefully).

Example: Let $f(x, y) = x^2 + y^2$. We want to minimize f subject to $6 - x - 2y \leq 0$. If we take $r_1 = 5$ in the definition of $\tilde{f}$ (so $\tilde{f}(x, y) = x^2 + y^2 - 5\ln(x + 2y - 6)$) and start with feasible point $(5, 5)$ we obtain a minimum at $(1.53, 3.05)$. Decreasing r_1 to 0.5 gives a minimum at $(1.24, 2.48)$, and decreasing r_1 to 0.05 gives a minimum at $(1.204, 2.408)$ (the true minimum is at $(1.2, 2.4)$).

One issue in using a barrier method is that of finding an initial feasible point which is in the interior of the feasible region. In many cases such a point will be obvious from considerations specific to the problem. If not, it can be rather difficult to find such a point (or perhaps prove that the feasible region is in fact empty if the constraints are mutually exclusive). One idea would be to use penalty functions, but on constraints $g_i(\mathbf{x}) \leq -\delta < 0$ with $f \equiv 0$. If a solution $\mathbf{a}$ to this can be found with $\tilde{f}(\mathbf{a}) = 0$ then $\mathbf{a}$ is a feasible point which is in the interior of the feasible region defined by $g_i(\mathbf{x}) \leq 0$.