

Operations Research Techniques and Algorithms (620-361)

Dr Yao-ban Chan

y.chan@ms.unimelb.edu.au

Telephone: 8344 9073

Office: Room 198, Richard Berry Building

Christina Burt

c.burt@ms.unimelb.edu.au

Telephone: 8344 1797

Office: 139 Barry St

Wednesday 30th April, 2008

Interpretation of KKT multipliers

KKT multipliers have the same physical interpretation as Lagrange multipliers — if the right-hand side of a constraint changes, then the optimal function value changes an amount proportional to the constraint change. Furthermore, the constant of proportionality is the KKT multiplier corresponding to the constraint.

We can see how this works for active and inactive inequality constraints. If an inequality constraint is inactive, its KKT multiplier must be 0. So changing the constraint does not change the optimal function value at all, which is as expected since it should not change the optimal solution!

On the other hand, active constraints have positive KKT multipliers. So if the right-hand side of a constraint with KKT multiplier λ^* changes by Δ , the optimal function value changes (approximately) by $-\lambda^* \Delta$.

In particular, if Δ is positive, we are enlarging the feasible region (recall that the constraint is $g(x) \leq 0$). Such a change must improve the optimal function value, which is why we have $\lambda^* \geq 0$ in the KKT conditions — if it were negative, the optimal function value would increase instead!

Today's Lecture

Constrained optimisation
Penalty methods

Penalty methods for nonlinear programs

In the previous section, we explored analytic methods for locating stationary points of constrained non-linear programs and the deciding whether these stationary points are local minima.

As we saw with both the single variable and multivariable unconstrained cases, it is quite often the case that it is difficult to apply these analytic methods. Usually this is because we cannot solve the equation $\nabla f = 0$ analytically or because, having solved this equation, it is difficult to show analytically that a second-order sufficient condition holds.

Because of this, we resorted to algorithmic methods for solving these problems. For single variable problems, we used the Fibonacci, Search, the Golden Section Search or Newton's method, while for multi-variable problems, we used the steepest descent algorithm, Newton's method or the BGFS algorithm.

It is even more difficult to solve the KKT criteria analytically to derive a stationary point and then verify the second order conditions for it to be a minimum when we have a constrained optimisation problem.

It is also more difficult to derive algorithmic methods for constrained problems.

One way of circumventing this is to “convert” a constrained problem to an unconstrained problem. This is usually achieved via the addition of a *penalty function* to the objective function.

The idea is to remove the constraints, but add a large penalty to the objective function, which comes into play if the constraints are not satisfied. The hope is that any local minimum of the modified objective function will be such that the constraints are satisfied.

The l_2 penalty method

We start by illustrating the use of l_2 -penalty methods for problems involving C^1 functions with an example.

Consider

$$\min_{x \in \mathcal{R}} f(x) := x^2 - 2x \quad \text{subject to} \quad g(x) = x \leq 0. \quad (1)$$

This has a unique (global) minimum, $x^* = 0$. We convert this to an unconstrained problem by choosing a *penalty parameter* $\alpha > 0$, and define the *penalty function*

$$P_\alpha(x) := x^2 - 2x + \frac{\alpha}{2}(x_+)^2,$$

where

$$x_+ := \max\{x, 0\} = \begin{cases} x & \text{if } x > 0 \quad (\text{infeasible}), \\ 0 & \text{if } x \leq 0 \quad (\text{feasible}). \end{cases}$$

$P_\alpha(x)$ is the objective function $f(x)$ plus the penalty term $(\alpha/2)(x_+)^2$. The penalty term measures the “cost” of infeasibility. The idea is that by letting α get large, in fact letting $\alpha \rightarrow \infty$, any infeasible point will become more costly than any feasible point, so that with “ $\alpha = \infty$ ”, minimizing $P_\alpha(x)$ is the same as solving the original problem.

For $\alpha > 0$, it can be shown that the function P_α is minimized at a point $x_\alpha > 0$, that satisfies

$$\begin{aligned} 0 &= \nabla P_\alpha(x_\alpha) = 2x - 2 + \alpha x \\ \implies x &= 2/(2 + \alpha). \end{aligned}$$

Thus $x_\alpha = 2/(2 + \alpha) \rightarrow 0 = x^*$ as $\alpha \rightarrow \infty$.

This example can be generalized to problems of the form (NLP).
For $\alpha > 0$, the l_2 penalty function for (NLP) is

$$P_\alpha(x) = f(x) + \frac{\alpha}{2} \left(\sum_i [g_i(x)_+]^2 + \sum_j h_j(x)^2 \right) \quad (2)$$

where

$$g_i(x)_+ := \max\{g_i(x), 0\} = \begin{cases} g_i(x) & \text{if } g_i(x) > 0 \quad (\text{infeasible}), \\ 0 & \text{if } g_i(x) \leq 0 \quad (\text{feasible}). \end{cases}$$

To see why the name “ l_2 ” is used, let $g(x)_+ := (g_1(x)_+, \dots, g_p(x)_+)$, then

$$\sum_i [g_i(x)_+]^2 + \sum_j h_j(x)^2 = \|g(x)_+\|^2 + \|h(x)\|^2,$$

That is, the penalty term uses the square of the Euclidean or l_2 norm.

The following convergence theorem shows the usefulness of the penalty function approach.

Theorem: Let f , g and h be C^1 functions. Suppose x^k minimizes P_{α_k} for each k , where $\alpha_k \rightarrow \infty$. If $\{x^k\}$ has a cluster point x^* and a constraint qualification holds at x^* , then x^* is (feasible and) stationary for (NLP).

Let's look at the first-order necessary condition for x^k to minimize P_α . Suppose f , g and h are C^2 functions.

Q: Is P_α C^1 ? Is it C^2 ?

A: From its definition, it can be shown that P_α is differentiable, and using the chain rule

$$\begin{aligned}\nabla P_\alpha(x) &= \nabla f(x) + \frac{\alpha}{2} \left(\sum_i \nabla_x [g_i(x)_+]^2 + \sum_j \nabla_x [h_j(x)]^2 \right) \\ &= \nabla f(x) + \alpha \left(\sum_i (g_i(x)_+) \nabla g_i(x) + \sum_j h_j(x) \nabla h_j(x) \right).\end{aligned}$$

Hence P_α is continuously differentiable.

However P_α is not C^2 because it is not generally twice differentiable if $g_i(x) = 0$ for some i .

For example, for $x \in \Re$, $(x_+)^2$ is C^1 but not C^2 at 0.

In the situation of the above theorem, suppose that $\{x^k\}$ converges to x^* . By choice of x^k , we have $0 = \nabla P_{\alpha_k}(x^k)$. Therefore if $\{\alpha_k g_i(x^k)_+\}_k$ is convergent for each i , that is $\{\alpha_k g(x^k)_+\}$ converges to some $\lambda^* \in \Re^p$; and if $\{\alpha_k h_j(x^k)\}_k$ is convergent for each j , that is $\{\alpha_k h(x^k)\}$ converges to some $\eta^* \in \Re^q$, then

$$\begin{aligned}0 &= \nabla P_{\alpha_k}(x^k) \\ &= \nabla f(x^k) + \sum_i (\alpha_k g_i(x^k)_+) \nabla g_i(x^k) + \sum_j (\alpha_k h_j(x^k)) \nabla h_j(x^k) \\ &\rightarrow \nabla f(x^*) + \sum_i \lambda_i^* \nabla g_i(x^*) + \sum_j \eta_j^* \nabla h_j(x^*).\end{aligned}$$

This is KKTa!

In other words, just as x^k approximates an optimal solution x^* of the constrained problem, $\alpha_k g_i(x^k)_+$ and $\alpha_k h_j(x^k)$ approximate the respective optimal multipliers λ_i^* and η_j^* .

In fact we can go a little bit further. Given $\lambda^* = \lim \alpha_k g(x^k)_+$ as above, the fact that $\alpha_k > 0$ and $g_i(x^k)_+ \geq 0$ by definition means that $\lambda_i^* \geq 0$. This is part of KKTb.

Also, if $x^k \rightarrow x^*$, and $g_i(x^*) < 0$, then for large enough k , $\alpha_k g_i(x^k)_+ = 0$ and therefore $\lambda_i^* = 0$. On the other hand if $\lambda_i^* > 0$ then $g_i(x^*)$ cannot possibly be strictly negative, and therefore $g_i(x^*) = 0$. This is another way of expressing $\lambda_i^* g_i(x^*) = 0$, which is also part of KKTb.

Example. Consider the nonlinear program

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 + x_1 - x_2 \\ \text{s.t.} \quad & x_1 \geq 1 \\ & x_2 \geq 0. \end{aligned}$$

1. Write down the l_2 -penalty function $P_k(x)$ with penalty parameter $\alpha_k = k$.
2. Find a stationary point $x^k = (x_1^k, x_2^k)$ for $P_k(x)$. Write down the limit $x^* = \lim_{k \rightarrow \infty} x^k$.
3. Write down an estimate of λ^k , the optimal multiplier vector, and find the limit $\lambda^* = \lim_{k \rightarrow \infty} \lambda^k$.



