Unit 12

Iterative Solution Algorithms for NLP without Restrictions

Let us remember that an **unconstrained NLPP** has an objective function that is being minimized, but does not have any constraints on what values the decision variables can take. This problem can be generically written as:

$$\begin{array}{ll} \min & f(x) \\ x \in \mathbb{R}^n, & \end{array}$$

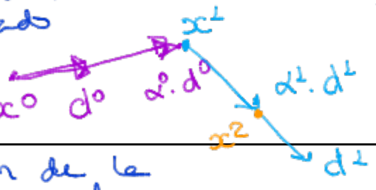
where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function.

Algorithm 1: Generic Algorithm for Unconstrained Nonlinear Optimization Problems

```
1 $k \leftarrow 0$
2 Fix $x^0 \in \mathbb{R}^n$        $T := \{0, 1, 2, \dots\}$
3 while Termination criteria are not met do
4     Find direction, $d^k$, to move in
5     Determine step size, $\alpha^k$
6     $x^{k+1} \leftarrow x^k + \underbrace{\alpha^k d^k}_{\text{el paso}}$  → actualización del estado
7     $k \leftarrow k + 1$  → actualización de la variable temporal
8 end
```

$x = (x_1, x_2, \dots, x_n)$

$d^k = (d^k_1, d^k_2, \dots, d^k_n)$



12.1 Search Direction

Because we focus on finding the search direction here, we ignore the step size parameter (i.e., α^k) in the following discussion. Thus, we assume in the following discussion that the points we generate after each iteration are defined as:

$$x^{k+1} = x^k + d^k.$$

After concluding the discussion of the search-direction methods here, we reintroduce the step size parameter in Section 12.3, where we focus exclusively on line search procedures.

12.1.1 Steepest Descent

$$\min_{d^k \in \mathbb{R}^n} \quad f(x^k + d^k).$$

$$\mathbb{R}^n \longrightarrow \mathbb{R}$$
$$d^k \mapsto f(\underline{\underline{x}}^k + d^k)$$

Since

$$f(x^k + d^k) \approx f(x^k) + {d^k}^\top \nabla f(x^k)$$

and the length of the d^k vector doesn't matter, we instead try to solve the optimization problem:

$$\begin{array}{ll} \min & f(x^k) + {d^k}^\top \nabla f(x^k) \\ \text{s.t.} & {d^k}^\top \cdot d^k = 1. \end{array}$$

Then, by the First-Order Necessary Condition (FONC) for Equality-Constrained Nonlinear Oprimization Problem, it holds:

$$\nabla f(x^k) + 2\lambda \cdot d^k = 0,$$

where λ is a Lagrange multiplier. And solving for d^k gives:

$$d^k = -\frac{1}{2 \cdot \lambda} \nabla f(x^k). \tag{12.1}$$

From this, we conclude that the solution in which $\lambda > 0$ is the minimum and from this observation we obtain:

Property 12.1 (Steepest Descent Rule)**.** Given an unconstrained nonlinear optimization problem and a current point, x^k, the **steepest descent** search direction is:

$$\boxed{d^k = -\nabla f(x^k).}$$

◊

Remark 12.2. The purpose of the $1/(2\lambda)$ term in Equation (12.1) is to scale the gradient so that the direction vector has a squared length of one. We can ignore this scaling, however, because the restriction that d^k have a squared length of one is arbitrarily added when we minimize the first-order Taylor approximation of $f(x^k+d^k)$. As discussed above, the length of the direction vector is ultimately unimportant because we always conduct a line search to determine how far to move in the search direction found. ◊

Example 12.3. See Example 5.1 of [1].

Prop Toda matriz simétrica tiene autovalores reales.

Prop Toda matriz simétrica es semidefinida positiva A ($A \succeq 0$) si y solo si todos sus autovalores ≥ 0.

Prop Toda matriz simétrica es definida positiva si y solo si todos sus autovalores son > 0.

Prop Si los autovalores de una matriz simétrica son todos > 0, entonces dicha matriz es invertible.

12.1.2 Newton's Method

Property 12.4 (Newton's Method Rule)**.** Given an unconstrained nonlinear optimization problem and a current point, x^k, the **Newton's method** search direction is:

$$d^k = -\left[\boxed{\nabla^2 f(x^k)}\right]^{-1}\nabla f(x^k),$$

so long as $\nabla^2 f(x^k)$ is invertible. Otherwise, the Newton's method search direction is not defined. ♢

Example 12.5. See Example 5.2 of [1].

$(\forall u \in \mathbb{R}^n : u^T A u \geq 0)$

$\forall u \in \mathbb{R}^n \setminus \{0\} : u^T A u > 0$

12.2 Termination Criteria

Both steepest descent and Newton's method use the same termination criteria, and that is to stop once reaching a stationary point.

Another issue that comes up in practice is that software packages typically numerically approximate partial derivatives when computing the gradient and Hessian. As such, the solver may be at a stationary point, but the computed gradient is not equal to zero due to approximation errors. Thus, most software packages include a parameter that allows some 'tolerance' in looking for a stationary point. For instance, the algorithm may terminate if

$$\|\nabla f(x^k)\| < \epsilon,$$

where ϵ is a user-specified tolerance.

Def Una matriz simétrica A es definida negativa ($A \prec 0$) si $-A$ es definida positiva.

12.3 Line Search

Once we determine a direction d^k in which to move, the next question is how far to move in the direction that is identified. This is the purpose of a 'line search'. We call this process a line search because what we are doing is 'looking' along a line in the search direction. As with search directions, there are many line-search methods available. We introduce three of them.

12.3.1 Exact Line Search/Line Minimization

An **exact line search** or **line minimization** solves the minimization problem:

$$\min_{\alpha^k \geq 0} \quad f(x^k + \alpha^k \cdot d^k),$$

to determine the optimal choice of step size. The advantage of an exact line search is that it provides the best choice of step size. This comes at a cost, however, which is that we must solve an optimization problem. This is always a single-variable problem, however, and in some instances the exact-line-search problem can be solved relatively easily. Indeed, the exact-line-search problem typically simplifies to solving:

$$\frac{\partial}{\partial \alpha^k} f(x^k + \alpha^k \cdot d^k) = 0,$$

because we can often treat the exact line search as being an unconstrained optimization problem. Nevertheless, an exact line search may be impractical for some problems.

Example 12.6. See Example 5.3 of [1].

12.3.2 Armijo Rule

The **Armijo rule** is a heuristic line search that is intended to give objective-function improvement without the overhead of having to solve an optimization problem, which is required in an exact line search. The Armijo rule works by specifying fixed scalars, s, β, and σ, with $s > 0$, $0 < \beta < 1$, and $0 < \sigma < 1$. We then have:

$$\alpha^k = \beta^{m^k} \cdot s,$$

comenzamos con $m^k \leftarrow 0$, i.e., $\alpha^k \leftarrow s$

where m^k is the smallest non-negative integer for which:

$$f(x^k) - f(x^k + \beta^{m^k} \cdot s \cdot d^k) \geq -\sigma \cdot \beta^{m^k} \cdot s \cdot {d^k}^\top \cdot \nabla f(x^k). \tag{12.2}$$

$\alpha^k \leftarrow \beta \cdot \alpha^k$

The idea behind the Armijo rule is that we start with an initial guess, s, of what a good step size is. If this step size gives us sufficient improvement in the objective function, which is measured by

$$f(x^k) - f(x^k + s \cdot d^k),$$

then we use this as our step size, i.e., we have $\alpha^k = s$. Otherwise, we reduce the step size by a factor of β and try this new reduced step size. That is, we check

$$f(x^k) - f(x^k + \beta \cdot s \cdot d^k).$$

We continue doing this, i.e., reducing the step size by a factor of β until we get sufficient improvement in the objective function. The term on the right-hand side of inequality (12.2) says that we want the objective to improve by some fraction of the direction times the gradient.

Usually, σ is chosen close to zero. Values in $[10^{-5}, 10^{-1}]$ are common. The reduction factor, β, is usually chosen to be between $1/2$ and $1/10$. The initial step size guess usually depends on what we know about the objective function. An initial guess of $s = 1$ can be used if we have no knowledge of the behavior of the objective function.

Example 12.7. See Example 5.4 of [1].

12.3.3 Pure Step Size

If we are using the Newton search direction, there is a third step size available to us, which is a **pure step size**. The pure step size rule is to simply fix

$$\alpha^k = 1.$$

It is important to stress that the pure step size should only be used with Newton's method. If it is used with steepest descent (or another search direction) there is no guarantee how the algorithm will perform.

12.4 Performance of Search Directions

A common question that arises is which of steepest descent or Newton's method is a 'better' search direction to use. The answer to this question is not a definite one, because they have advantages and disadvantages relative to one another. First of all, it is clear that Newton's method involves more work in each iteration of the Generic Algorithm for Unconstrained Nonlinear Optimization Problems; however, in many instances Newton's method provides greater improvement in the objective function in each iteration.

Example 12.8. See Example 5.5 of [1].

In this example, both methods provide the same amount of objective-function improvement in one iteration. This means that the added work involved in computing the Newton's method direction is wasted. However, the following example shows that this does not hold true for all problems.

Example 12.9. See Example 5.6 of [1].

In this example, Newton's method performs much better than steepest descent, finding a stationary point and terminating after a single iteration. To understand why this happens, let us analyze the contour plots of the objective functions...

Generally speaking, how well steepest descent performs in one iteration depends on the ratio between the largest and smallest eigenvalues of the Hessian matrix. The closer this ratio is to one, the better steepest descent performs. As this ratio gets higher, steepest descent tends to perform worse. Newton's method uses second derivative information, which is encoded in the Hessian matrix, to mitigate these scaling issues.

Although Newton's method handles poorly scaled problems better than steepest descent, it has drawbacks (in addition to it requiring more work per iteration). This is illustrated in the following example.

Example 12.10. See Example 5.7 of [1].

Local maxima and saddle points are also stationary points and Newton's method attempts to find a stationary point, without distinguishing between what type of stationary point it is. So, in order to determine a priori whether Newton's method will move in a direction that gives an objective-function improvement or not, let us we examine a first-order Taylor approximation of the objective-function value after conducting a Newton iteration...

If the matrix is positive definite, the algorithm uses the Newton's method direction. Otherwise, another search direction (such as steepest descent) is used.

Unlike Newton's method, steepest descent is guaranteed to give an objective-function improvement in each iteration, so long as the step size is chosen appropriately. We can see this from the first-order Taylor approximation of the objective function after conducting an iteration...

Thus, steepest descent is an example of what is known as a **descent algorithm**: the objective function is guaranteed to improve at each iteration of the Generic Algorithm for Unconstrained Nonlinear Optimization Problems.

On its own, Newton's method has no such guarantee. If, however, we use Newton's method only at points where the Hessian is positive definite and use steepest descent at other points, we will also have a descent algorithm. This hybrid algorithm will also only stop at stationary points that are local minima or saddle points (so long as it does not start at a local maximum).

12.5 Exercises

Exercises 5.1, 5.2 and 5.3 of [1] (see Section 5.8).

Bibliography

[1] Sioshansi, R., and Conejo, A. J. *Optimization in Engineering: Models and Algorithms.* Springer Optimization and Its Applications. Springer, 2017.