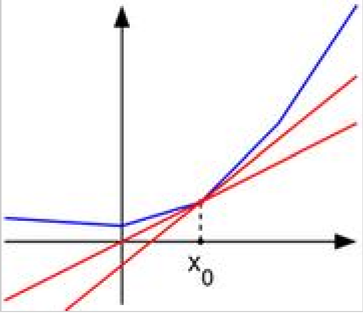
Subgradient optimization

From optimization

Author Name: Aaron Anderson (ChE 345 Spring 2015)
Steward: Dajun Yue and Fengqi You

Subgradient Optimization (or **Subgradient Method**) is an iterative algorithm for minimizing convex functions, used predominantly in Nondifferentiable optimization for functions that are convex but nondifferentiable. It is often slower than Newton's Method when applied to convex differentiable functions, but can be used on convex nondifferentiable functions where Newton's Method will not converge. It was first developed by Naum Z. Shor in the Soviet Union in the 1960's.



A convex nondifferentiable function (blue) with red "subtangent" lines generalizing the derivative at the nondifferentiable point x_0.

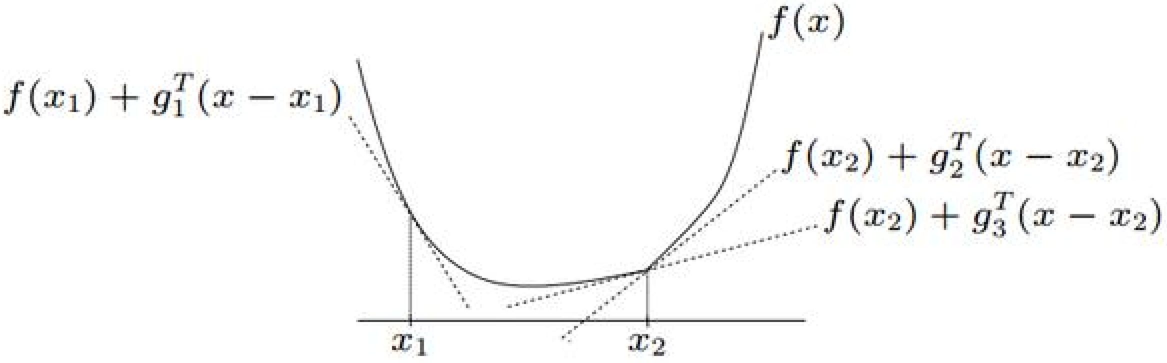
Contents

Introduction

The **Subgradient** (related to Subderivative and Subdifferential) of a function is a way of generalizing or approximating the derivative of a convex function at nondifferentiable points. The definition of a subgradient is as follows: g is a subgradient of f at x if, for all y, the following is true:

$$f(y) \geq f(x) + g^T(y - x)$$

An example of the subgradient of a nondifferentiable convex function f can be seen below:


$f(x)$
$f(x_1) + g_1^T(x - x_1)$
$f(x_2) + g_2^T(x - x_2)$
$f(x_2) + g_3^T(x - x_2)$
x_1
x_2

Where g_1 is a subgradient at point x_1 and g_2 and g_3 are subgradients at point x_2. Notice that when the function is differentiable, such as at point x_1, the subgradient, g_1, just becomes the gradient to the function. Other important factors of the subgradient to note are that the subgradient gives a linear global underestimator of f and if f is convex, then there is at least one subgradient at every point in its domain. The set of all subgradients at a certain point is called the subdifferential, and is written as $\partial f(x_0)$ at point x_0.

The Subgradient Method

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with domain $\mathbb{R}^n$. To minimize f the subgradient method uses the iteration:

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$$

Where k is the number of iterations, $x^{(k)}$ is the kth iterate, $g^{(x)}$ is *any* subgradient at $x^{(k)}$, and $\alpha_k\ (> 0)$ is the kth step size. Thus, at each iteration of the subgradient method, we take a step in the direction of a negative subgradient. As explained above, when f is differentiable, $g^{(k)}$ simply reduces to $\nabla\ f(x^{(k)})$. It is also important to note that the subgradient method is not a descent method in that the new iterate is not always the best iterate. Thus we need some way to keep track of the best solution found so far, *i.e.* the one with the smallest function value. We can do this by, after each step, setting

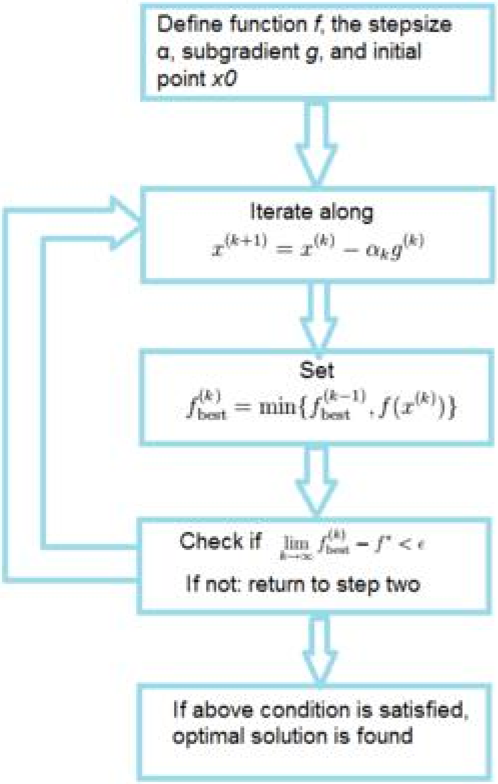
$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(x^{(k)})\}$$

and setting $i_{\text{best}}^{(k)} = k$ if $x^{(k)}$ is the best (smallest) point found so far. Thus we have:

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \ldots, f(x^{(k)})\}$$

which gives the best objective value found in k iterations. Since this value is decreasing, it has a limit (which can be $-\infty$).

An algorithm flowchart is provided below for the subgradient method:


Define function *f*, the stepsize α, subgradient *g*, and initial point *x0*
Iterate along $x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$
Set $f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(x^{(k)})\}$
Check if $\lim_{k\to\infty} f_{\text{best}}^{(k)} - f^* < \epsilon$
If not: return to step two
If above condition is satisfied, optimal solution is found

Step size

Several different step size rules can be used:

- **Constant step size**: $\alpha_k = h$ independent of k.
- **Constant step length**: $\alpha_k = h/\|g^{(k)}\|_2$. This means that $\|x^{(k+1)} - x^{(k)}\|_2 = h$.
- **Square summable but not summable**: These step sizes satisfy

  $$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  One typical example is $\alpha_k = a/(b+k)$, where $a > 0$ and $b \geq 0$.

- **Nonsummable diminishing**: These step sizes satisfy

  $$\lim_{k\to\infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  One typical example is $\alpha_k = a/\sqrt{k}$, where $a > 0$.

An important thing to note is that for all four of the rules given here, the step sizes are determined "off-line", or before the method is iterated. Thus the step sizes do not depend on preceding iterations. This "off-line" property of subgradient methods differs from the "on-line" step size rules used for descent methods for differentiable functions where the step sizes do depend on preceding iterations.

Convergence Results

There are different results on convergence for the subgradient method depending on the different step size rules applied. For constant step size rules and constant step length rules the subgradient method is guaranteed to converge within some range of the optimal value. Thus:

$$\lim_{k\to\infty} f_{\text{best}}^{(k)} - f^{\star} < \epsilon$$

where f^* is the optimal solution to the problem and ϵ is the aforementioned range of convergence. This means that the subgradient method finds a point within ϵ of the optimal solution f^*. ϵ is number that is a function of the step size parameter h , and as h decreases the range of convergence ϵ also decreases, *i.e.* the solution of the subgradient method gets closer to f^* with a smaller step size parameter h . For the diminishing step size rule and the square summable but not summable rule, the algorithm is guaranteed to converge to the optimal value or $\lim_{k\to\infty} f(x^{(k)}) = f^{\star}$. When the function f is differentiable the subgradient method with constant step size yields convergence to the optimal value, provided the parameter h is small enough.

Example: Piecewise linear minimization

Suppose we wanted to minimize the following piecewise linear convex function using the subgradient method:

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i).$$

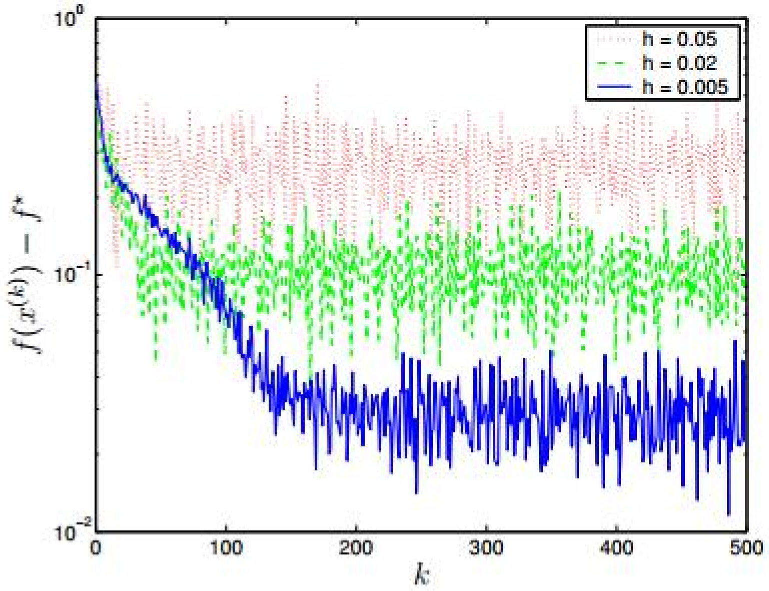
Since this is a linear programming problem finding a subgradient is simple: given x we can find an index j for which:

$$a_j^T x + b_j = \max_{i=1,\ldots,m}(a_i^T x + b_i).$$

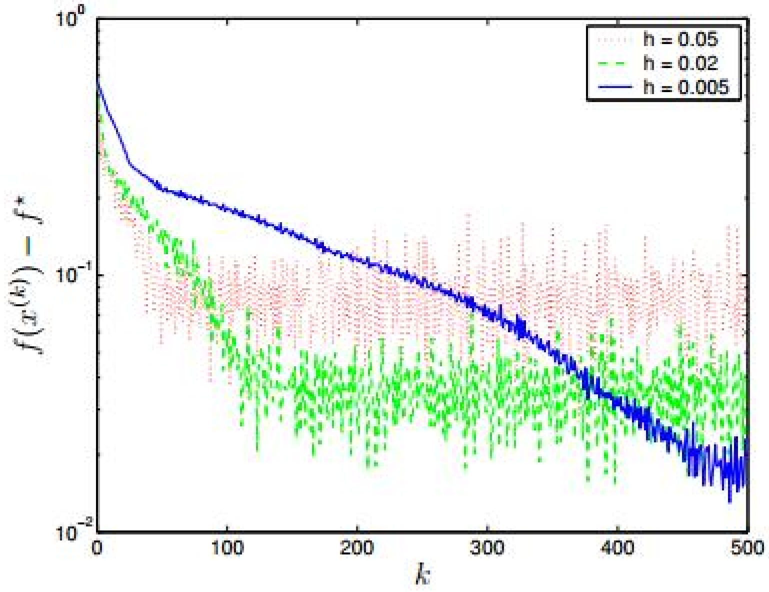
The subgradient in this case is $g = a_j$. Thus the iterative update is then:

$$x^{(k+1)} = x^{(k)} - \alpha_k a_j$$

Where j is chosen such to satisfy $a_j^T x^{(k)} + b_j = \max_{i=1,\ldots,m}(a_i^T x^{(k)} + b_i)$. In order to apply the subgradient method to this problem all that is needed is some way to calculate $\max_i(a_i^T x + b_i)$ and the ability to carry out the iterative update. Even if the problem is dense and very large (where standard linear programming might fail), if there is some efficient way to calculate f then the subgradient method is a reasonable choice for algorithm. Consider a problem with $n = 10$ variables and $m = 100$ terms and with data a_i and b_i generated from a normal distribution. We will consider all four of the step size rules mentioned above and will plot ϵ or the difference between the optimal solution and the subgradient solution as a function of k, the nuber of iterations. For the constant step size rule $\alpha_k = h$, for several values of h the following plot was obtained:

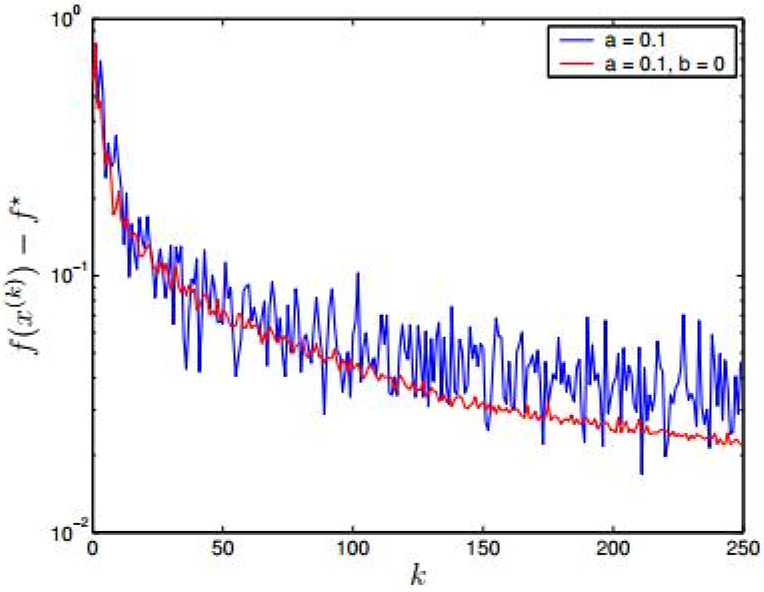


For the constant step length rule $\alpha_k = h/\|g^{(k)}\|_2$ for several values of h the following plot was obtained:



The above figures reveal a trade-off: a larger step size parameter h gives a faster convergence but in the end gives a larger range of suboptimality so it is important to determine an h that will converge close to the optimal solution without taking a very large number of iterations.

For the subgradient method using diminishing step size rules, both the nonsummable diminishing step size rule $\alpha = 0.1/\sqrt{k}$ (blue) and the square summable but not summable step size rule $\alpha = 0.1/k$ (red) are plotted below for convergence:



This figure illustrates that both the nonsummable diminishing step size rule and the square summable but not summable step size rule show relatively fast and good convergence. The square summable but not summable step size rule shows less variation than the nonsummable diminishing step size rule but both show similar speed and convergence.

Overall, all four step size rules can be used to get good convergence, so it is important to try different values for h in the constant step size and length rules and different formulas for the nonsummable diminishing step size rule and the square summable but not summable step size rule in order to get good convergence in the smallest amount of iterations possible.

Conclusion

The subgradient method is a very simple algorithm for minimizing convex nondifferentiable functions where newton's method and simple linear programming will not work. While the subgradient method has a disadvantage in that it can be much slower than interior-point methods such as Newton's method, it as the advantage of the memory requirement being often times much smaller than those of an interior-point or Newton method, which means it can be used for extremely large problems for which interior-point or Newton methods cannot be used. Morever, by combining the subgradient method with primal or dual decomposition techniques, it is sometimes possible to develop a simple distributed algorithm for a problem. The subgradient method is therefor an important method to know about for solving convex minimization problems that are nondifferentiable or very large.

References

1. Akgul, M. "Topics in Relaxation and Ellipsoidal Methods", volume 97 of Research Notes in Mathematics. Pitman, 1984.
2. Bazaraa, M. S., Sherali, H. D. "On the choice of step size in subgradient optimization." European Journal of Operational Research 7.4, 1981
3. Bertsekas, D. P. "Nonlinear Programming", (2nd edition), Athena Scientific, Belmont, MA, 1999.
4. Goffin, J. L. "On convergence rates of subgradient optimization methods." Mathematical Programming 13.1, 1977.
5. Shor, N. Z. "Minimization Methods for Non-differentiable Functions". Springer Series in Computational

Mathematics. Springer, 1985.
6. Shor, N. Z. "Nondifferentiable Optimization and Polynomial Problems". Nonconvex Optimization and its Applications. Kluwer, 1998.