Abstract

In our work, we explore the Steepest Descent method, which could be seen as an variation of the well-studied Gradient Descent method. First, we will give the exact definition of the Steepest Descent method, then we will give convergence analysis for both these two methods, which are very similar.
Then, through our experiment, we showed the drastic impact on the performance of the Steepest Descent method by the choice norm we use. Finally, we try to use the Hessian to automatically construct the norm, which we put the G-S descent, but sadly failed.

1 INTRODUCTION

In the field of solving unconstrained optimization problem, one most well-known and well-studied method, is the Gradient Descent method, we first briefly introduce the Gradient Descent, which is shown in Algorithm 1.
In the step 2 of the Gradient Descent method, we must choose exact line search or backtracking line search, while exact line search is preferred, in our experiments we will use the backtracking line search, which is more easy to implement. The backtracking line search is shown in Algorithm 2.

Algorithm 1 Gradient descent method.

Given a starting point $x \in \mathbf{dom} f$
repeat
1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$

until stopping criterion is satisfied.

Algorithm 2 Backtracking line search.

Given a starting point $\triangle x$ for f at $x \in \mathbf{dom} f, \alpha \in (0, 0.5), \beta \in (0, 1)$
While $f(x + t \triangle x) > f(x) + \alpha t \nabla f(x)^T \triangle x$
$t = \beta t$

We have to mention due to some historical reasons the Gradient Descent is also called the Steepest Descent method[1](Well, its direction is truly locally 'steepest'). However, the Steepest Descent method is another story.
Let $||\cdot||$ be any norm on R^n. We define a normalized steepest descent direction(with respect to the norm $||\cdot||$) as

$$\Delta x_{nsd} = argmin\{\nabla f(x)^T v | ||v|| = 1\}$$

Now we can give the Steepest descent method in Algorithm 3. To get a better understanding of the Steepest Descent Method, let's look at some examples.

Algorithm 3 Steepest descent method

Given a starting point $x \in \mathbf{dom} f$
repeat
1. Compute steepest descent direction Δx_{nsd}.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$

until stopping criterion is satisfied.

1.1 Steepest Descent for Euclidean and quadratic norms

If we take the norm $||\cdot||$ to be the Euclidean norm, then the steepest descent direction is simply the negative gradient.
Given a $P \in S^n_{++}$, now we consider the quadratic norm

$$||z||_P = (z^T P z)^{1/2} = ||P^{1/2}||_2$$

The normalized steepest descent direction is given by

$$\Delta x_{nsd} = -(\nabla f(x)^T P^{-1} \nabla f(x))^{-1/2} P^{-1} \nabla f(x)$$

2 Discussion and examples

2.1 choice of norm for steepest descent

One important thing for the steepest descent method is to choose a good norm. In our convergence analysis, we know that the gradient method works well when the condition number of the sublevel sets are moderate, and poorly otherwise. Let's begin with the quadratic P-norm, we know that using the quadratic P-norm means that we change coordinates by $\overline{x} = P^{1/2}x$. So, a good idea would be to make the ellipsoid

$$\varepsilon = \{x | x^T P x \leq 1\}$$

be a good approximation of the shape of the sublevel set.(In other words, it gives a good approximation after appropriate scaling and translation.)
Let's consider the following function :

$$g_1(x_1, x_2) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1}$$

For all three tries, we start at point$(-0.4, 1)$, and use the backtracking line search with $\alpha = 0.2$, $\beta = 0.8$, and let the initial step be 1, the tolerance is 0.01. For the two steepest descent tries, we choose $P_1 = \begin{matrix} 1/2 & 0 \\ 0 & 1/4 \end{matrix}$ and $P_2 = \begin{matrix} 1/4 & 0 \\ 0 & 1/2 \end{matrix}$, the results are shown in Figure **??**.
We also add the shape of the norms on the graphs. From the results, we can clearly see that the norm P_1, which is a better approximate of the sublevel set, give the best result. And we we use P_2, it becomes worse than the original

S-descent with P_1 : 6 iterations

S-descent with P_1 : 10 iterations

G-S-descent with Hessian : 19 iterations

Figure 1: Two methods for g_3.

gradient descent.

Sadly, the G-S only slow downs the process, which is not hard to understand, when the x converges to the optima, the direction of the gradient becomes stable, so the norm won't do any good. So, if we choose to use the Steepest Descent, we should use the norm we choose from the beginning.

3 CONCLUSIONS

4 REFERENCES

Nonlinear Programming: Analysis and Methods Dover Publishing
Convex Optimization Stephen Boyd and Lieven Vandenberghe