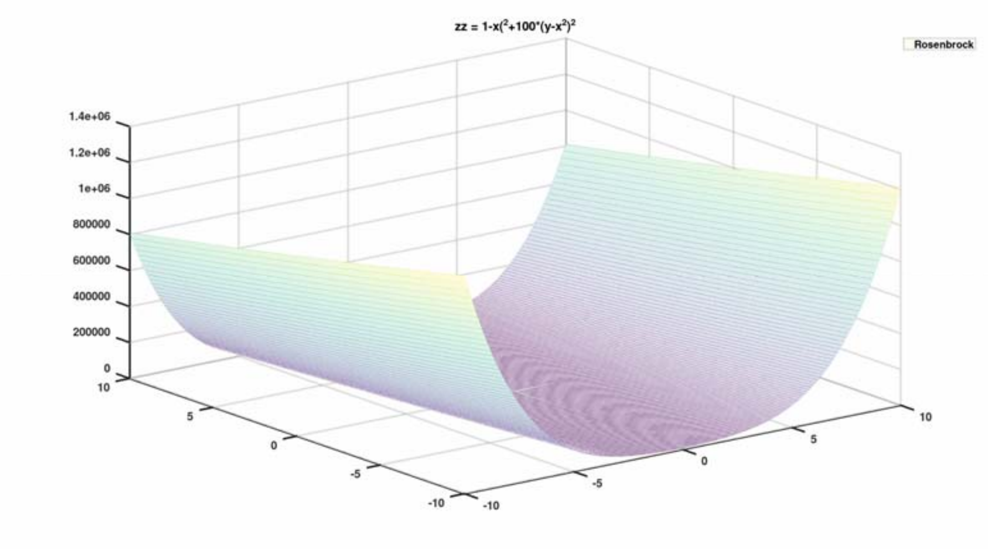
Bief Note on Optimization 04@20240911

©DuyKy Nguyen, PhD EE @ Unitethc.com 2024-10-22

We're using Rosenbrock to evaluate methods performane as usual in the Optimization literature



1. Multivariate Calculus

$f(\mathbf{x}) \in \Re, \mathbf{x} \in \Re^n$

Gradient $\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$

Hessian $\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_2} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{bmatrix}$ is symmetric square matrix nxn

For $\mathbf{f}(\mathbf{x}) \in \Re^m,\ \mathbf{x} \in \Re^n$ we have Jacobianmatrix mxn

Jacobian $\mathbf{J}f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial^2 x_1} & \frac{\partial^2 f_1}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 x_n} \\ \frac{\partial^2 f_2}{\partial x_2 x_2} & \frac{\partial^2 f_2}{\partial^2 x_2} & \cdots & \frac{\partial^2 f_n}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_m}{\partial x_n x_1} & \frac{\partial^2 f_m}{\partial x_n x_2} & \cdots & \frac{\partial^2 f_m}{\partial^2 x_n} \end{bmatrix}$

2. Line Search

Basic therr're 2 types: exact and inexact
Line search plays a key role in method performance

2.1. Exact Line search

It's the simplest but working fine with **simple function** like $\boxed{z = x^2 + 3y^2}$ but not **Rosenbrock function** $\boxed{z = (x-1)^2 + 100(y - x^2)^2}$

Tthe search direction often has the form $\min_{a>0}[f(\mathbf{x}_k + a\mathbf{p}_k)]$

or using univariate function

$$\phi(a) = f(\mathbf{x}_k + a\mathbf{p}_k) \approx f(\mathbf{x}_k) + a\mathbf{p}_k\nabla^T f(\mathbf{x}_k) + \tfrac{1}{2}a^2\mathbf{p}_k\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k^T$$

$$\min \textit{when with}\, \mathbf{g}_k = \nabla f(\mathbf{x}_k),\, \mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$$

$$\phi'(a) = \mathbf{p}_k\nabla^T f(\mathbf{x}_k) + a^2\mathbf{p}_k\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k^T = \mathbf{p}_k\mathbf{g}_k^T + a\mathbf{p}_k\mathbf{H}_k\mathbf{p}_k^T = 0$$

hence

$$a = \frac{-\mathbf{p}_k\mathbf{g}_k^T}{\mathbf{p}_k\mathbf{H}_k\mathbf{p}_k^T}, \mathbf{g}_k = \nabla f(\mathbf{x}_k), \mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$$

basically it has a form below

$$\min_{a>0}[f(\mathbf{x}_k + a\mathbf{p}_k)]$$

o rusing univariate function

$$\phi(a) = f(\mathbf{x}_k + a\mathbf{p}_k) \approx f(\mathbf{x}_k) + a\mathbf{p}_k\nabla^T f(\mathbf{x}_k) + \tfrac{1}{2}a^2\mathbf{p}_k\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k^T$$

$$\min \textit{when with}\, \mathbf{g}_k = \nabla f(\mathbf{x}_k),\, \mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$$

$$\phi'(a) = \mathbf{p}_k\nabla^T f(\mathbf{x}_k) + a^2\mathbf{p}_k\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k^T = \mathbf{p}_k\mathbf{g}_k^T + a\mathbf{p}_k\mathbf{H}_k\mathbf{p}_k^T = 0$$

hence

$$a = \frac{-\mathbf{p}_k\mathbf{g}_k^T}{\mathbf{p}_k\mathbf{H}_k\mathbf{p}_k^T}, \mathbf{g}_k = \nabla f(\mathbf{x}_k), \mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$$

$$\boxed{a = \frac{-\mathbf{p}_k\mathbf{g}_k^T}{\mathbf{p}_k\mathbf{H}_k\mathbf{p}_k^T}, \mathbf{g}_k = \nabla f(\mathbf{x}_k), \mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)}$$

2.2. Inexact Line search

Used in both sime descent method and BFGS method using complicated approx of Hessian as difference of gradient for search direction conjugate gradient for faster convergence but it works only incase startpoint close to solution point where the simple descent method still get to solution

Rosenbrock functio has solution point at point **(1,1)**

BFGS gets to solution from initial poin**t -2, -2)** fut fails from far point**(-200, -200)** while the simple descent thod smethod still work from that far polintr

Basically it uses descent gradient $-\nabla f(\mathbf{x})$ to reduce function value $f(\mathbf{x})$ in an ieration loop with initial step $a = 1$ and update using Wolfe condition 1 W-1 [**11-45**]

$$\boxed{f(\mathbf{x}_k + a_k\mathbf{d}_k) \le f(\mathbf{x}_k) + a_k\beta_1\nabla^T f(\mathbf{x}_k)\mathbf{d}_k, \beta_1 \in (0,1)} \quad [a]$$

The Wolfe condition 2 W-2 is [**11-47**] ratio of new derivative/old<bb2

$$\boxed{\frac{\nabla^T f(\mathbf{x}_k + a_k\mathbf{d}_k)\mathbf{d}_k}{\nabla^T f(\mathbf{x}_k)\mathbf{d}_k} \le \beta_2, \ < 0 < \beta_1 < \beta_2 < 1}$$

2.3. Algorithm in Algorithm 11.5: Line search

from Optimization:Principles and Algorithms by
Michel Bierlaire http://optimizationprinciplesalgorithms.com

Objective

To find a step a such that the Wolfe conditions (W-1) and (W-2) are satisfied.

Input

The continuously differentiable function f : Rn → R.

The gradient of the function ∇ f : Rn → Rn.

A vector x ∈ Rn.

A descent direction d such that ∇ f(x)Td < 0.

An initial solution $\alpha_0 > 0$ (e.g. $\alpha_0 = 1$). $a_0 = 1$

Parameters β_1 and β_2 such that $0 < \beta_1 < \beta_2 < 1$ (e.g., $\beta_1 = 10^{-4}$ and $\beta_2 = 0.99$).

A parameter $\lambda > 1$ (e.g., $\lambda = 2$).

Output

A step $a*$ such that the conditions (W-1) and (W-2) are satisfied.

Initialization

$i := 0$.

$a_{lo} = 0$.

a

$a_{hi} = +\infty$

Repeat

if a_i violates (W.1) then the step is **too long**

$a_{hi} = a_i$

$a_{i+1} = \frac{a_{lo} + a_{hi}}{2}$

if a_i a_i does not violate (1W-11.45) but violates (W-211.47) then the step is **too short**

$a_{lo} = a_i$

if $a_{hi} < \infty$ then $a_{i+1} = \frac{a_{lo} + a_{hi}}{2}$

else $a_{i+1} = \lambda a_i$

Until both W-1 and W-2 satisfied

We haveBy Taylor expansion

$f(\mathbf{x}_k + a_k \mathbf{d}_k) \approx f(\mathbf{x}_k) + a_k \mathbf{d}_k \nabla^T f(\mathbf{x}_k)$

So the Wolfe cond reduces function

3. Algorithm 13.1: Quasi-Newton BFGS method

from Optimization:Principles and Algorithms by Michel Bierlaire http://optimizationprinciplesalgorithms.com

Just for completeness and curity

but it's too complicated to use in realitydue to accumulated err

Objective

To find (an approximation of) a local minimum of the problem

$\min_{x \in R^n} f(x)$

Input

The continuously differentiable function f : Rn → R.

The gradient n- row vector of n $\mathbf{g} = \nabla f(\mathbf{x})$

Inital point $\mathbf{x}_0$

Initial approx **Hessian inverse** $\tilde{\mathbf{H}} = \mathbf{I}$
Required tolerance $\varepsilon > 0$

Output

Solution pt $\mathbf{x}^*$

Init

k=0

Repeat

$\mathbf{d}_k = -\tilde{\mathbf{H}}_k \nabla f(\mathbf{x}_k)$

step length a by line search $\mathbf{x}_{k+1} = \mathbf{x}_k + a\mathbf{d}_k$

$$\tilde{\mathbf{H}}_{k=} = \left(\mathbf{I} - \frac{\mathbf{d}_{k-1} - \mathbf{y}_{k-1}^T}{\mathbf{d}_{k-1}^T \mathbf{y}_{k-1}}\right) \tilde{\mathbf{H}}_{k-1} \left(\mathbf{I} - \frac{\mathbf{y}_{k-1} - \mathbf{d}_{k-1}^T}{\mathbf{d}_{k-1}^T \mathbf{y}_{k-1}}\right) + \frac{\mathbf{d}_{k-1}\mathbf{d}_{k-1}^T}{\mathbf{d}_{k-1}^T \mathbf{y}_{k-1}}$$

Update *with*

$$\mathbf{d}_{k-1} = a_{k-1}\mathbf{d}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$$
$$\mathbf{y}_{k-1} = \nabla f(\mathbf{x}_x) - \nabla f(\mathbf{x}_{x-1})$$

Until $\|\nabla f(\mathbf{x}_k)\| < \varepsilon$

SOLUTION $\mathbf{x}^* = \mathbf{x}_k$

4. Conclusion

Linesearch function hasembedded gradient a=L_S(x) to provide step size yo move tany initial point $x_0$to solution point

UseThe descent method using line search id the most reliable with simple nalgorithm below

1. Start point row vector $\mathbf{x}_0$
2. Get search dir $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ In **BFGS** $\mathbf{d}_k = -\tilde{\mathbf{H}}_k \nabla f(\mathbf{x}_k)$
3. Get step a by linesearch a=**L_S(x_k, d_k)**
4. **Get next point** $\mathbf{x}_{k+1} = \mathbf{x}_k + a\mathbf{d}$
5. Repeat 2 to 2 untril $\|\nabla f(\mathbf{x}_k)\| < \varepsilon$ or max iteration reaches