Nonlinear Optimization

Maestría en Cómputo Estadístico

Centro de Investigación en Matemáticas A.C.

What is nonlinear optimization? I

- A nonlinear optimization problem is also referred to as a "nonlinear programming"
- It consists of the optimization of a function subject to constraints, such that any function can be nonlinear
- Linear optimization is a special case of nonlinear optimization

What is nonlinear optimization? II

- Nonlinear optimization models arise often in science and engineering
- Some remarkable examples:
  - Support vector machines
  - Neural network training
  - Shape optimization
  - Image reconstruction

Optimality Conditions I

- The nonlinear optimization methods that we will discuss are not always guaranteed to find a global minimum
- These algorithms terminate at a stationary point which usually is at least a local minimum
- For unconstrained nonlinear optimization, there are two conditions that, when satisfied, imply that an optimal solution has been found

CIMAT

Optimality Conditions II

- A well-known result is that an objective function f defined and differentiable, a point $\mathbf{x}^*$ is a local maximum or minimum if its gradient is zero
- The above is known as the *first order optimality condition*, and can be formally stated as:

First order optimality condition

Lef $f : \mathcal{X} \to \mathbb{R}$ be a function defined on a set $\mathcal{X} \subseteq \mathbb{R}^n$. Suppose that $\mathbf{x}^* \in \mathcal{X}$ is a local optimum point and that the partial derivatives of f exist at $\mathbf{x}^*$, then $\nabla f(\mathbf{x}^*) = 0$

CIMAT

Optimality Conditions III

- A second condition uses the second derivative and is stated as

Second order optimality condition

Lef $f : \mathcal{X} \to \mathbb{R}$ be a function defined on a set $\mathcal{X} \subseteq \mathbb{R}^n$. Suppose that f is twice continuously differentiable over $\mathcal{X}$ and that $\mathbf{x}^*$ is a stationary point, then

- if $\mathbf{x}^*$ is a local minimum of f over $\mathcal{X}$, then $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite
- if $\mathbf{x}^*$ is a local maximum of f over $\mathcal{X}$, then $\nabla^2 f(\mathbf{x}^*)$ is negative semi-definite

CIMAT

Steepest descent method Method for Minimization I

- The Steepest descent method method is a classical method for nonlinear optimization
- It is one of the simplest and the most fundamental minimization methods for unconstrained optimization
- It makes use of the first-order necessary condition for a local minimizer

$$\nabla f(\mathbf{x}) = 0 \quad (1)$$

Steepest descent method Method for Minimization II

- The main idea is to consider the search as an interative procedure
- An initial search point is required and is updated in each iteration based on a step size, α
- The direction is given by $\nabla f\left(\mathbf{x}\right)$, and the step size determines how far we go in that particular direction
- A point is updated as follows:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f\left(\mathbf{x}^{(t)}\right) \tag{2}$$

CIMAT

Steepest descent method Method for Minimization III

Algorithm 1 Steepest descent

1. Set an initial search point $\mathbf{x}^{(0)}$
2. Set a step size, $\alpha \geq 0$
3. Set $t \leftarrow 0$
4. **while** $\| \nabla f \left(\mathbf{x}^{(t)}\right) \| > \epsilon$ **do**
5. $\quad \mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \alpha \nabla f \left(\mathbf{x}^{(t)}\right)$
6. $\quad t \leftarrow t + 1$
7. **end while**
8. **return** $\mathbf{x}^{(t)}$

Example I

- Minimize $f(\mathbf{x}) = \frac{3}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1x_2 - 2x_1$. Assume that the initial point is $\mathbf{x}^{(0)} = (-2, 4)$ and the step size is 0.5

Solution:

1. $\nabla f(\mathbf{x}) = (3x_1 - x_2 - 2, x_2 - x_1)$
2. $\nabla f\left(\mathbf{x}^{(0)}\right) = (-12, 6)$
   $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - 0.5\nabla f\left(\mathbf{x}^{(0)}\right)$
   $\mathbf{x}^{(1)} = (-2 + 6, 4 - 3) = (4, 1)$

Example II

3. $\nabla f\left(\mathbf{x}^{(1)}\right) = (9.000, -3.000)$
   $\mathbf{x}^{(2)} = (4.000 - 4.500, 1.000 + 1.500) = (-0.500, 2.500)$
4. $\nabla f\left(\mathbf{x}^{(2)}\right) = (-6.000, 3.000)$
   $\mathbf{x}^{(2)} = (-0.500 + 3.000, 2.500 - 1.500) = (2.500, 1.000)$
5. $\nabla f\left(\mathbf{x}^{(3)}\right) = (4.500, -1.500)$
   $\mathbf{x}^{(4)} = (2.500 - 2.250, 1.000 + 0.750) = (0.250, 1.750)$
6. $\nabla f\left(\mathbf{x}^{(4)}\right) = (-3.000, 1.500)$
   $\mathbf{x}^{(5)} = (0.250 + 1.500, 1.750 - 0.750) = (1.750, 1.000)$
7. $\nabla f\left(\mathbf{x}^{(5)}\right) = (2.250, -0.750)$
   $\mathbf{x}^{(6)} = (1.750 - 1.125, 1.000 + 0.375) = (0.625, 1.375)$

CIMAT

Example III

- Repeat the process with:
  - Step size equals to 1.0
  - Step size equals to 0.1

Steepest Descent

- In this method, the convergence strongly depends on the step size
- The optimal step size can be determined by solving

$$\min f\left(\mathbf{x}^{(t)} - \alpha \nabla f\left(\mathbf{x}^{(t)}\right)\right) \tag{3}$$

- Repeat the previous exercise computing the optimal step size in each iteration

The Newton's Method I

- The main idea is to start with an arbitrary point, and iteratively try to find the point at which the function evaluates to zero
- Ergo, it iteratively uses the quadratic approximation of the objective function
- An initial solution is required and updated in each iteration as follows:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{\nabla f\left(\mathbf{x}^{(t)}\right)}{H_f\left(\mathbf{x}^{(t)}\right)} \tag{4}$$

The Newton's Method II

Algorithm 2 Newton's Method

1: Set an initial search point $\mathbf{x}^{(0)}$
2: Set $t \leftarrow 0$
3: **while** $\| \nabla f\left(\mathbf{x}^{(t)}\right) \| > \epsilon$ **do**
4: $\quad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \dfrac{\nabla f\left(\mathbf{x}^{(t)}\right)}{H_f\left(\mathbf{x}^{(t)}\right)}$
5: $\quad t \leftarrow t + 1$
6: **end while**
7: **return** $\mathbf{x}^{(t)}$

CIMAT

Example I

- Minimize $f(\mathbf{x}) = \frac{3}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1x_2 - 2x_1$. Assume that the initial point is $\mathbf{x}^{(0)} = (-2, 4)$

Solution:

- Gradient:

$$\nabla f(\mathbf{x}) = (3x_1 - x_2 - 2, x_2 - x_1) \tag{5}$$

- Hessian matrix:

$$\begin{bmatrix} 3 & -1 \\ -1 & 1 \end{bmatrix} \tag{6}$$

CIMAT

Example II

Solution:

1. $\nabla f\left(\mathbf{x}^{(0)}\right) = (-12, 6)$
2. $\mathbf{x}^{(2)} = (-2.000 + 3.00, 4.000 - 3.000) = (1.000, 1.000)$
3. $\nabla f\left(\mathbf{x}^{(1)}\right) = (0, 0)$

CIMAT

Exercise

Solve the following optimization problems using both Steepest Descend and Newton's Method

1. $x_1^4 + 2x_1x_2 + x_2^4$
2. $x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$
3. $2x_1^2 + x_2^2 - 2x_1x_2 + 2x_1^3 + x_1^4$

CIMAT

Conjugate Gradient Descent I

- Conjugate gradient descent can be regarded as being between the method of steepest descent and Newton's method
- It aims at accelerating the convergence of steepest descent while avoids the computational cost of the Newton's method
- Unlike previous methods, in conjugate gradient descent, at each iteration the direction is modified by a combination of the earlier directions

Conjugate Gradient Descent II

Algorithm 3 Conjugate Gradient Descent Method

Set an initial search point $\mathbf{x}^{(0)}$
Set a step size, $\alpha \geq 0$
Set $t \leftarrow 0$
$s(t) \leftarrow -\nabla f\left(\mathbf{x}^{(t)}\right)$
$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \alpha s(t)$
while $\| \nabla f\left(\mathbf{x}^{(t)}\right) \| > \epsilon$ **do**
    $t \leftarrow t + 1$
    $s(t) = -\nabla f\left(\mathbf{x}^{(t)}\right) + \dfrac{\| \nabla f\left(\mathbf{x}^{(t)}\right) \|^2}{\| \nabla f\left(\mathbf{x}^{(t-1)}\right) \|^2} s(t-1)$
    $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \alpha s(t)$
end while
return $\mathbf{x}^{(t)}$

CIMAT

Example I

- Minimize $f(\mathbf{x}) = \frac{3}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1x_2 - 2x_1$. Assume that the initial point is $\mathbf{x}^{(0)} = (-2, 4)$ and $\alpha = 0.5$

Solution:

- Gradient:

$$\nabla f(\mathbf{x}) = (3x_1 - x_2 - 2, x_2 - x_1) \tag{7}$$

1. $\nabla f\left(\mathbf{x}^{(0)}\right) = (-12, 6)$
   $s(0) = (12, -6)$
   $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha s(0) = (4, 1)$
2. $\nabla f\left(\mathbf{x}^{(1)}\right) = (9, -3)$
   $s(1) = (-9, 3) + \frac{1}{2}(12, -6) = (-3, 0)$
   $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha s(1) = (2.5, 1.0)$

CIMAT

Example II

3. $\nabla f\left(\mathbf{x}^{(2)}\right) = (4.500, -1.500)$

   $s(2) = (-4.500, 1.500) + \frac{1}{4}(-3.000, 0.000) = (-5.250, 1.500)$

   $\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha s(2) = (-0.125, 1.750)$

4. $\nabla f\left(\mathbf{x}^{(3)}\right) = (-4.125, 1.875)$

   $s(3) = (4.125, -1.875) + \frac{657}{720}(-5.250, 1.500) = (-0.6666, -0.506)$

   $\mathbf{x}^{(4)} = \mathbf{x}^{(3)} + \alpha s(3) = (-0.458, 1.497)$

Conjugate Gradient Descent: Variants I

The conjugate gradient direction can be rewritten as:

$$s(t) = -\nabla f\left(\mathbf{x}^{(t)}\right) + \beta s(t-1) \tag{8}$$

Some variants are listed below:

- **Fletcher–Reeves**:

$$\beta = \frac{\nabla f\left(\mathbf{x}^{(t)}\right)^T \nabla f\left(\mathbf{x}^{(t)}\right)}{\nabla f\left(\mathbf{x}^{(t-1)}\right)^T \nabla f\left(\mathbf{x}^{(t-1)}\right)} \tag{9}$$

Conjugate Gradient Descent: Variants II

- **Polak–Ribière**:

$$\beta = \frac{\nabla f\left(\mathbf{x}^{(t)}\right)^T \left(\nabla f\left(\mathbf{x}^{(t)}\right) - \nabla f\left(\mathbf{x}^{(t-1)}\right)\right)}{\nabla f\left(\mathbf{x}^{(t-1)}\right)^T \nabla f\left(\mathbf{x}^{(t)}\right)} \tag{10}$$

- **Hestenes-Stiefel**:

$$\beta = \frac{\nabla f\left(\mathbf{x}^{(t)}\right)^T \left(\nabla f\left(\mathbf{x}^{(t)}\right) - \nabla f\left(\mathbf{x}^{(t-1)}\right)\right)}{s(t-1)^T \left(\nabla f\left(\mathbf{x}^{(t)}\right) - \nabla f\left(\mathbf{x}^{(t-1)}\right)\right)} \tag{11}$$

Quasi-Newton methods I

- Quasi-Newton methods, as the name suggest, are an alternative to the Newton's method
- They are specially useful when computing the Hessian matrix is computational prohibitive
- They rank among the most efficient methods available

Quasi-Newton methods II

- The basic principle in quasi-Newton methods is that the direction of search is based on an $n \times n$ direction matrix $\mathbf{S}$ that is generated from available data and is contrived to be an approximation of $\mathbf{H}^{-1}$
- Quasi-Newton methods updates the "Hessian matrix" by analyzing successive gradient vectors

Quasi-Newton methods III

- In general, a point is updated as follows:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \mathbf{S}^{(\mathbf{t})} \nabla f(\mathbf{x}^{(t)}) \tag{12}$$

where

$$\mathbf{S}^{(\mathbf{t})} = \begin{cases} \mathbf{I}_n & \text{for the steepest descent method} \\ \mathbf{H}_{\mathbf{t}}^{-1} & \text{for the Newton method} \end{cases}$$

- The idea is, therefore, to choose some positive definite $\mathbf{S}^{(\mathbf{t})}$ such that is equal to or, at least, approximately equal to $\mathbf{H}_{\mathbf{t}}^{-1}$

Quasi-Newton methods IV

Algorithm 4 Basic Quasi-Newton Method

```
Set $t \leftarrow 0$
Set an initial search point $\mathbf{x}^{(t)}$
Set a step size, $\alpha \geq 0$
Set $\mathbf{S}^{(\mathbf{t})} \leftarrow \mathbf{I}_n$
while $\| \nabla f \left(\mathbf{x}^{(t)}\right) \| > \epsilon$ do
    $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \mathbf{S}^{(\mathbf{t})} \nabla f(\mathbf{x}^{(t)})$
    Update $\mathbf{S}^{(\mathbf{t+1})}$ by following a proper strategy
    $t \leftarrow t + 1$
end while
return $\mathbf{x}^{(t)}$
```

Newton's Method vs Quasi-Newton Methods

Quasi-Newton Method	Newton's Method
Only need the function values and and gradients	Need the function values, gradients and Hessians
H_t^{-1} maintains positive definite for several updates	H_t^{-1} is not sure to be positive definite
Need $\mathcal{O}(n^2)$ multiplications in each iteration	Need $\mathcal{O}(n^3)$ multiplications in each iteration

CIMAT

How can we update the Hessian Matrix? I

There are several approaches to approximate the Hessian matrix

- **Symmetric Rank-One**:

$$H_{t+1} = H_t + \frac{\left(y_{(t)} - H_t s_{(t)}\right)\left(y_{(t)} - H_t s_{(t)}\right)^T}{\left(y_{(t)} - H_t s_{(t)}\right)^T s_{(t)}} \tag{13}$$

where:

$$y_{(t)} = \nabla f\left(\mathbf{x}^{(t)}\right) - \nabla f\left(\mathbf{x}^{(t-1)}\right)$$
$$s_{(t)} = \mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}$$

CIMAT

How can we update the Hessian Matrix? II

- **Davidon–Fletcher–Powell (DFP)**:

$$H_{t+1} = H_t + \frac{s_{(t)} s_{(t)}^T}{s_{(t)}^T y_{(t)}} - \frac{\left(H_t y_{(t)}\right)\left(H_t y_{(t)}\right)^T}{y_{(t)}^T H_{(t)} y_{(t)}} \tag{14}$$

- **Broyden–Fletcher–Goldfarb–Shanno (BFGS)**:

$$H_{t+1} = H_t + \frac{y_{(t)} y_{(t)}^T}{y_{(t)}^T s_{(t)}} - \frac{H_t s_{(t)} s_{(t)}^T H_{(t)}^T}{s_{(t)}^T H_{(t)} s_{(t)}} \tag{15}$$

Example I

- Minimize $f(\mathbf{x}) = \frac{3}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1x_2 - 2x_1$. Assume that the initial point is $\mathbf{x}^{(0)} = (-2, 4)$ and $\alpha = 0.5$

Solution:

1. Initial Hessian matrix
$$H_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
2. The gradient vector: $\nabla f(\mathbf{x}) = (3x_1 - x_2 - 2, x_2 - x_1)$
3. $\nabla f\left(\mathbf{x}^{(0)}\right) = (-12, 6)$
$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha H_0 \nabla f\left(\mathbf{x}^{(0)}\right) = (4, 1)$

Example II

4. $\nabla f\left(\mathbf{x}^{(1)}\right) = (9.000, -3.000)$
   $S_0 = \mathbf{x}^{(1)} - \mathbf{x}^{(0)} = (6.000, -3.000)$
   $y_0 = \nabla f\left(\mathbf{x}^{(1)}\right) - \nabla f\left(\mathbf{x}^{(0)}\right) = (21.000, -9.000)$

$$H_1 = H_0 + \frac{(y_0 - H_0 s_0)(y_0 - H_0 s_0)^T}{(y_0 - H_0 s_0)^T s_0}$$
$$= \begin{bmatrix} 3.083 & -0.833 \\ -0.833 & 1.333 \end{bmatrix}$$

5. $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \alpha H_1^{-1} \nabla f\left(\mathbf{x}^{(1)}\right) = (2.610, 1.256)$

CIMAT

Example III

6. $\nabla f\left(\mathbf{x}^{(2)}\right) = (4.573, -1.354)$
   $S_0 = \mathbf{x}^{(2)} - \mathbf{x}^{(1)} = (-1.390, 0.256)$
   $y_0 = \nabla f\left(\mathbf{x}^{(2)}\right) - \nabla f\left(\mathbf{x}^{(1)}\right) = (-4.427, 1.646)$

$$H_2 = H_1 + \frac{(y_1 - H_1 s_1)(y_1 - H_1 s_1)^T}{(y_1 - H_1 s_0)^T s_1}$$
$$= \begin{bmatrix} 3.000 & -1.000 \\ -1.000 & 1.000 \end{bmatrix}$$

7. $\mathbf{x}^{(3)} = \mathbf{x}^{(2)} - \alpha H_2^{-1} \nabla f\left(\mathbf{x}^{(2)}\right) = (1.805, 1.128)$

Quasi-Newton Methods

- Note, however, that the above approaches still require computing the inverse of the Hessian matrix
- Can we instead update the inverse of the Hessian matrix?
- The BFGS formula can be updated as follows:

$$H_{t+1}^{-1} = \left(\mathbf{I}_n - \frac{s_t y_t^T}{s_t^T y_t}\right) H_t^{-1} \left(\mathbf{I}_n - \frac{y_t s_t^T}{s_t^T y_t}\right) + \frac{s_t s_t^T}{s_t^T y_t} \tag{16}$$

CIMAT

Questions?