Non-Linear Optimization Methods - Part I

Optimization Methods in Management Science
Master in Management
HEC Lausanne

Dr. Rodrigue Oeuvray

Fall 2019 Semester

Non-Linear Optimization Methods

Unconstrained optimization:

- Quadratic programming
  - Direct resolution
  - Conjuguate gradients

Differentiable Optimization

- When the objective function is **smooth**, numerical methods using information about **derivatives** (first order and possibly second order) perform better than derivative-free algorithms

- In the coming slides, we will focus on **quadratic programming**. For a quadratic function, it is **easy** to compute its gradient and its hessian

- Quadratic programming has many applications. Two applications will be discussed in this course: **Portfolio Optimization** and **Support Vector Machine**

Quadratic Programming

- Suppose we want to solve the following problem $\mathcal{Q}$:

$$min_{\boldsymbol{x} \in \mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + c,$$

  where $\boldsymbol{Q}$ is a **symmetric positive definite** matrix

- They are many different methods to solve this problem. We present here a direct resolution of it
- The solution of this system is simply given by (first order conditions):

$$\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{g} = \boldsymbol{0} \iff \boldsymbol{Q}\boldsymbol{x} = -\boldsymbol{g} \iff \boldsymbol{x} = -\boldsymbol{Q}^{-1}\boldsymbol{g}$$

- **Solving an unconstrained quadratic problem defined by a symmetric positive definite matrix is equivalent to solving a linear system !**

Direct Resolution of a Quadratic Program (1)

- A direct resolution is typically based on the Cholesky decomposition of matrix $\boldsymbol{Q}$
- A Cholesky decomposition of a **symmetric positive definite** matrix $\boldsymbol{Q}$ is defined as $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix
- A square matrix is called lower triangular if all the entries above the main diagonal are zero
- **This decomposition only works when the matrix is positive definite !**

Cholesky Decomposition : Example

- We would like to the determine the Cholesky decomposition of $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 4 & 2 \\ 3 & 2 & 11 \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix}$$

$$= \begin{pmatrix} L_{11}^2 & * & * \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & * \\ L_{11}L_{31} & L_{21}L_{31} + L_{22}L_{32} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}$$

- By identification, it is easy to determine the values of the L_{ij}:
  - row 1, column 1: $L_{11} = 1$
  - row 2, column 1: $L_{21} = 0$
  - row 2, column 2: $L_{22} = 2$
  - ...

Cholesky Decomposition : Example (Cont'd)

We finally get the following decomposition :

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 4 & 2 \\ 3 & 2 & 11 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & 0 & 0 \\ \mathbf{0} & \mathbf{2} & 0 \\ \mathbf{3} & \mathbf{1} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 0 & \mathbf{2} & \mathbf{1} \\ 0 & 0 & \mathbf{1} \end{pmatrix}$$

Direct Resolution of a Quadratic Program (2)

A **direct resolution** is based on the following steps:

(1) Perform a Cholesky decomposition of $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix

(2) Compute $\boldsymbol{y}$ the solution of $\boldsymbol{L}\boldsymbol{y} = -\boldsymbol{g}$

(3) Compute $\boldsymbol{x}$ the solution of $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{y}$

Then $\boldsymbol{x}$ is the **unique** minimizer of the problem

Example

- We consider the following problem:

$$min_{\boldsymbol{x} \in \mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x}$$

where $\boldsymbol{Q}$ and $\boldsymbol{g}$ are given by:

$$\boldsymbol{Q} = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

- One can check that $\boldsymbol{Q}$ is a symmetric positive definite matrix
- The Cholesky decomposition of $\boldsymbol{Q}$ is

$$\boldsymbol{Q} = \begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix}$$

Example (Cont'd)

First, we need to solve :

$$\begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \iff \left\{ \begin{array}{ccccccc} 2y_1 & & & & & = & -1 \\ 6y_1 & + & y_2 & & & = & -1 \\ -8y_1 & + & 5y_2 & + & 3y_3 & = & -1 \end{array} \right.$$

We get that $y_1 = -0.5, y_2 = 2,$ and $y_3 = -5$

Example (Cont'd)

Then, we need to find $\boldsymbol{x}$ such that

$$\begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -0.5 \\ 2 \\ -5 \end{pmatrix} \Longleftrightarrow \left\{ \begin{array}{ccccccc} 2x_1 & + & 6x_2 & - & 8x_3 & = & -0.5 \\ & & x_2 & + & 5x_3 & = & 2 \\ & & & & 3x_3 & = & -5 \end{array} \right.$$

We conclude that the unique minimizer of this problem is given by:

$$x_1 = -\frac{455}{12}, x_2 = \frac{31}{3}, x_3 = -\frac{5}{3}$$

Quadratic Programming: The Conjuguate Gradient Method

- Another approach to solve an **unconstrained** quadratic problem $f(\boldsymbol{x})$ when the matrix is **positive definite** is based on an **iterative** method called conjuguate gradient

- It generates a **sequence of iterates** converging to the **optimal** solution of the problem

- The algorithm **stops** when a **stopping criterion** is met. It can be a **maximum number** of iterations and/or a criterion based on some necessary conditions to get an optimum (typically $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$, where ϵ is a small positive number

The Conjugate Gradient Method

- Let $\boldsymbol{Q}$ be a $n \times n$ **positive definite** matrix. We say that the non-null vectors $\boldsymbol{d}_k$ are $\boldsymbol{Q}$-conjuguate if $\boldsymbol{d}_i^T \boldsymbol{Q} \boldsymbol{d}_j = 0, \forall i, j$ such that $i \neq j$

- The idea is to generate a sequence of iterates $\boldsymbol{x}_k$ converging to the optimal solution:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \quad k = 1, \ldots, n,$$

  with $\alpha_k = argmin_\alpha f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$

- $\alpha_k (\in \mathbb{R})$ is the **optimal** scalar that minimizes the objective function in the direction given by $\boldsymbol{d}_k$. Such α_k is determined by the following formula:

$$\alpha_k = -\frac{\boldsymbol{d}_k^T (\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{g})}{\boldsymbol{d}_k^T \boldsymbol{Q} \boldsymbol{d}_k}$$

The Conjugate Gradient Method (Cont'd)

The directons $\boldsymbol{d}_i$ are computed based on the **Gram-Schmidt process** applied to $-\nabla f(\boldsymbol{x}_1), \ldots, -\nabla f(\boldsymbol{x}_i)$:

$$\boldsymbol{d}_i = -\nabla f(\boldsymbol{x}_i) + \sum_{k=1}^{i-1} \frac{\boldsymbol{d}_k^T \boldsymbol{Q} \nabla f(\boldsymbol{x}_i)}{\boldsymbol{d}_k^T \boldsymbol{Q} \boldsymbol{d}_k} \boldsymbol{d}_k$$

Indeed, one can directly compute $\boldsymbol{d}_i$ from $\boldsymbol{d}_{i-1}$ with the following formula :

$$\boldsymbol{d}_i = -\boldsymbol{Q}\boldsymbol{x}_i - \boldsymbol{g} + \beta_i \boldsymbol{d}_{i-1},$$

where

$$\beta_i = \frac{\nabla f(\boldsymbol{x}_i)^T \nabla f(\boldsymbol{x}_i)}{\nabla f(\boldsymbol{x}_{i-1})^T \nabla f(\boldsymbol{x}_{i-1})} = \frac{(\boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{g})^T (\boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{g})}{(\boldsymbol{Q}\boldsymbol{x}_{i-1} + \boldsymbol{g})^T (\boldsymbol{Q}\boldsymbol{x}_{i-1} + \boldsymbol{g})}$$

The CG Algorithm (1)

We consider an **unconstrained quadratic minimization** problem given by a **symmetric positive definite** matrix $\boldsymbol{Q}$ and a vector $\boldsymbol{g}$

- **Input**: a first approximation $\boldsymbol{x}_1$ of the solution, a convergence parameter ϵ, and a parameter k counting the number of iterations
- **Output**: the (approximated) solution $\boldsymbol{x}^*$ of the problem
- **Stopping criteria**:
  - If $k < n$ and $\|\nabla f(\boldsymbol{x}_{k+1})\| < \epsilon$, then $\boldsymbol{x}^* = \boldsymbol{x}_{k+1}$
  - If $k = n$, then $\boldsymbol{x}^* = \boldsymbol{x}_{n+1}$

The CG Algorithm (2)

- **Initialization**: $k = 1, \boldsymbol{d}_1 = -\boldsymbol{Q}\boldsymbol{x}_1 - \boldsymbol{g}$
- **Iterations**:
  - Compute the step α_k:

$$\alpha_k = -\frac{\boldsymbol{d}_k^T(\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{g})}{\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k}$$

  - Compute the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$
  - Check the stopping criterion. If it is satisfied, then stop: $\boldsymbol{x}_{k+1}$ is the (approximated) solution of the problem. If not, then continue the current iteration
  - Compute β_{k+1}:

$$\beta_{k+1} = \frac{\nabla f(\boldsymbol{x}_{k+1})^T \nabla f(\boldsymbol{x}_{k+1})}{\nabla f(\boldsymbol{x}_k)^T \nabla f(\boldsymbol{x}_k)} = \frac{(\boldsymbol{Q}\boldsymbol{x}_{k+1} + \boldsymbol{g})^T(\boldsymbol{Q}\boldsymbol{x}_{k+1} + \boldsymbol{g})}{(\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{g})^T(\boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{g})}$$

  - Compute the new direction $\boldsymbol{d}_{k+1} = -\boldsymbol{Q}\boldsymbol{x}_{k+1} - \boldsymbol{g} + \beta_{k+1}\boldsymbol{d}_k$
  - k = k + 1

Convergence of the CG Algorithm

- It converges in **at most** n iterations !
- For a quadratic problem defined by a symmetric positive definite matrix, a **sufficient** condition to have a minimum is that the gradient at that point should be **null**
- If the gradient is sufficiently close to $\mathbf{0}$ (the null vector), then we consider that we have found the optimum
- Concretely, if the **norm** of gradient evaluated at the current iterate is sufficiently small ($\|\nabla f(\boldsymbol{x}_{k+1})\| < \epsilon$), then we consider that **the algorithm has converged** to the optimum

Example

We consider the following problem:

$$min_{\boldsymbol{x} \in \mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x}$$

where $\boldsymbol{Q}$ and $\boldsymbol{g}$ are given by:

$$\boldsymbol{Q} = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} \text{-1} \\ \text{-2} \end{pmatrix}$$

Example (Cont'd)

- Solving this problem is equivalent to solving the following system:

$$\begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- A direct resolution (with Gauss elimination or with the Cholesky decomposition) gives the following solution $\boldsymbol{x}^* \approx (0.09\ 0.64)^T$

- Rather that solving directly this problem, let's try to compute the steps generated by the CG algorithm

- We assume that the starting point (chosen arbitrarily) for the CG algorithm is $\boldsymbol{x}_1 = (2\ 1)^T$

Example: Iteration 1

- Initialization: $k = 1$, $\epsilon = 10^{-6}$,

$$\boldsymbol{d}_1 = -\boldsymbol{Q}\boldsymbol{x}_1 - \boldsymbol{g} = -\begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} -1 \\ -2 \end{pmatrix} = \begin{pmatrix} -8 \\ -3 \end{pmatrix}$$

- Compute the step α_1:

$$\alpha_1 = -\frac{\boldsymbol{d}_1^T(\boldsymbol{Q}\boldsymbol{x}_1 + \boldsymbol{g})}{\boldsymbol{d}_1^T\boldsymbol{Q}\boldsymbol{d}_1} = -\frac{\begin{pmatrix} -8 & -3 \end{pmatrix}\begin{pmatrix} 8 \\ 3 \end{pmatrix}}{\begin{pmatrix} -8 & -3 \end{pmatrix}\begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} -8 \\ -3 \end{pmatrix}} = \frac{73}{331}$$

- Compute the next iterate $\boldsymbol{x}_2$:

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + \alpha_1\boldsymbol{d}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} + \frac{73}{331}\begin{pmatrix} -8 \\ -3 \end{pmatrix} \approx \begin{pmatrix} 0.2356 \\ 0.3384 \end{pmatrix}$$

Example: Iteration 1 (Cont'd)

- For the stopping criterion, we need to compute $\|\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g}\|$:

$$\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g} = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} 0.2356 \\ 0.3384 \end{pmatrix} + \begin{pmatrix} -1 \\ -2 \end{pmatrix} \approx \begin{pmatrix} 0.2810 \\ -0.7492 \end{pmatrix}$$

  As $\|\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g}\| = 0.80 > \epsilon$, we continue the current iteration

- Compute β_2:
  - From the previous calculations: $\boldsymbol{Q}\boldsymbol{x}_1 + \boldsymbol{g} = (8\ 3)^T$
  - Then:

$$\beta_2 = \frac{(\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g})^T(\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g})}{(\boldsymbol{Q}\boldsymbol{x}_1 + \boldsymbol{g})^T(\boldsymbol{Q}\boldsymbol{x}_1 + \boldsymbol{g})} = \frac{0.6401}{73} \approx 0.0088$$

- Compute the new direction $\boldsymbol{d}_2$:

$$\boldsymbol{d}_2 = -\boldsymbol{Q}\boldsymbol{x}_2 - \boldsymbol{g} + \beta_2 \boldsymbol{d}_1 = \begin{pmatrix} -0.2810 \\ 0.7492 \end{pmatrix} + \beta_2 \begin{pmatrix} -8 \\ -3 \end{pmatrix} = \begin{pmatrix} -0.3511 \\ -0.7229 \end{pmatrix}$$

Example: Iteration 2

- $k = 2$
- We compute α_2:

$$\alpha_2 = -\frac{\boldsymbol{d}_2^T(\boldsymbol{Q}\boldsymbol{x}_2 + \boldsymbol{g})}{\boldsymbol{d}_2^T\boldsymbol{Q}\boldsymbol{d}_2} = 0.4122$$

- Next iterate $\boldsymbol{x}_3$:

$$\boldsymbol{x}_3 = \boldsymbol{x}_2 + \alpha_2\boldsymbol{d}_2 = \begin{pmatrix} 0.2356 \\ 0.3384 \end{pmatrix} + 0.4122 \begin{pmatrix} -0.3511 \\ -0.7229 \end{pmatrix} = \begin{pmatrix} 0.0909 \\ 0.6364 \end{pmatrix}$$

- Stopping criterion: $k = 2$ (= the dimension of the problem) and the process is **completed**

The CG algorithm has found the solution of this linear system in two iterations ! This point is the unique solution of our problem