Numerical Methods I
Mathematical Programming

Aleksandar Donev
Courant Institute, NYU[1]
donev@courant.nyu.edu

[1]Course G63.2010.001 / G22.2420-001, Fall 2010

October 21st, 2010

Outline

1. Fundamentals
2. Smooth Unconstrained Optimization
3. Constrained Optimization
4. Conclusions

Mathematical Programming

- The general term used is **mathematical programming**.
- Simplest case is **unconstrained optimization**

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

where $\mathbf{x}$ are some variable parameters and $f:\ \mathbb{R}^n \to \mathbb{R}$ is a scalar **objective function**.

  - Find a **local minimum** $\mathbf{x}^\star$:

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \quad \text{s.t.} \quad \|\mathbf{x}-\mathbf{x}^\star\| \leq R > 0.$$

  (think of finding the bottom of a valley).
  - Find the best local minimum, i.e., the **global minimumx***: This is virtually impossible in general and there are many specialized techniques such as **genetic programming**, **simmulated annealing**, **branch-and-bound** (e.g., using interval arithmetic), etc.
- Special case: A **strictly convex objective function** has a unique local minimum which is thus also the global minimum.

Constrained Programming

- The most general form of **constrained optimization**

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$$

  where $\mathcal{X} \subset \mathbb{R}^n$ is a **set of feasible solutions**.

- The feasible set is usually expressed in terms of **equality and inequality constraints**:

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$
$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$$

- The only generally solvable case: **convex programming**
  Minimizing a convex function $f(\mathbf{x})$ over a convex set $\mathcal{X}$: every local minimum is global.
  If $f(\mathbf{x})$ is strictly convex then there is a **unique local and global minimum**.

Special Cases

- Special case is **linear programming**:

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left\{\mathbf{c}^T\mathbf{x}\right\}$$
$$\text{s.t.}\qquad \mathbf{A}\mathbf{x}\leq\mathbf{b}\qquad .$$

- Example from homework 4 (now online!) is equality-constrained **quadratic programming**

$$\min_{\mathbf{x}\in\mathbb{R}^2}\left\{x_1^2+x_2^2\right\}$$
$$\text{s.t.}\quad x_1^2+2x_1x_2+3x_2^2=1\quad .$$

generalized to arbitary ellipsoids as:

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left\{f(\mathbf{x})=\|\mathbf{x}\|_2^2=\mathbf{x}\cdot\mathbf{x}=\sum_{i=1}^n x_i^2\right\}$$
$$\text{s.t.}\qquad (\mathbf{x}-\mathbf{x}_0)^T\mathbf{A}(\mathbf{x}-\mathbf{x}_0)=1\qquad .$$

Necessary and Sufficient Conditions

- **First-order necessary condition** for a local minimizer is that $\mathbf{x}^{\star}$ be a **critical point (maximum, minimum or saddle point)**:

$$\mathbf{g}\left(\mathbf{x}^{\star}\right)=\nabla_{\mathbf{x}} f\left(\mathbf{x}^{\star}\right)=\left\{\frac{\partial f}{\partial x_i}\left(\mathbf{x}^{\star}\right)\right\}_i=\mathbf{0},$$

  and a **second-order necessary condition** is that the Hessian is positive **semi-definite**,

$$\mathbf{H}\left(\mathbf{x}^{\star}\right)=\nabla^2_{\mathbf{xx}} f\left(\mathbf{x}^{\star}\right)=\left\{\frac{\partial^2 f}{\partial x_i \partial x_j}\left(\mathbf{x}^{\star}\right)\right\}_{ij}\succeq\mathbf{0}.$$

- A **second-order sufficient condition** for a critical point $\mathbf{x}^{\star}$ to be a local minimum if the Hessian is **positive definite,**

$$\mathbf{H}\left(\mathbf{x}^{\star}\right)=\nabla^2_{\mathbf{xx}} f\left(\mathbf{x}^{\star}\right)\succ\mathbf{0}$$

  which means that the minimum really looks like a valley or a **convex** bowl.

Descent Methods

- Finding a local minimum is generally **easier** than the general problem of solving the non-linear equations

$$\mathbf{g}\left(\mathbf{x}^{\star}\right) = \boldsymbol{\nabla}_{\mathbf{x}} f\left(\mathbf{x}^{\star}\right) = \mathbf{0}$$

  - We can evaluate f in addition to $\boldsymbol{\nabla}_{\mathbf{x}} f$.
  - The Hessian is positive-(semi)definite near the solution (enabling simpler linear algebra such as Cholesky).

- If we have a current guess for the solution $\mathbf{x}^k$, and a **descent direction** (i.e., **downhill** direction) $\mathbf{d}^k$:

$$f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right) < f\left(\mathbf{x}^k\right) \text{ for all } 0 < \alpha \leq \alpha_{max},$$

  then we can move downhill and get closer to the minimum (valley):

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k,$$

  where $\alpha_k > 0$ is a **step length**.

Gradient Descent Methods

- For a differentiable function we can use Taylor's series:

$$f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right) \approx f\left(\mathbf{x}^k\right) + \alpha_k \left[\left(\boldsymbol{\nabla} f\right)^T \mathbf{d}^k\right]$$

- This means that **fastest local decrease** in the objective is achieved when we move opposite of the gradient: **steepest or gradient descent**:

$$\mathbf{d}^k = -\boldsymbol{\nabla} f\left(\mathbf{x}^k\right) = -\mathbf{g}_k.$$

- One option is to choose the step length using a **line search** one-dimensional minimization:

$$\alpha_k = \arg\min_{\alpha} f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right),$$

which needs to be solved **only approximately**.

Steepest Descent

- Assume an exact line search was used, i.e., $\alpha_k = \arg\min_\alpha \phi(\alpha)$ where

$$\phi(\alpha) = f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right).$$

$$\phi'(\alpha) = 0 = \left[\nabla f\left(\mathbf{x}^k + \alpha \mathbf{d}^k\right)\right]^T \mathbf{d}^k.$$

- This means that steepest descent takes a **zig-zag path** down to the minimum.
- Second-order analysis shows that steepest descent has **linear convergence** with convergence coefficient

$$C \sim \frac{1-r}{1+r}, \quad \text{where} \quad r = \frac{\lambda_{min}(\mathbf{H})}{\lambda_{max}(\mathbf{H})} = \frac{1}{\kappa_2(\mathbf{H})},$$

inversely proportional to the **condition number** of the Hessian.
- Steepest descent can be very slow for ill-conditioned Hessians: One improvement is to use **conjugate-gradient method instead** (see book).

Newton's Method

- Making a second-order or quadratic model of the function:

$$f(\mathbf{x}^k + \Delta\mathbf{x}) = f(\mathbf{x}^k) + \left[\mathbf{g}\left(\mathbf{x}^k\right)\right]^T(\Delta\mathbf{x}) + \frac{1}{2}(\Delta\mathbf{x})^T\left[\mathbf{H}\left(\mathbf{x}^k\right)\right](\Delta\mathbf{x})$$

  we obtain **Newton's method**:

$$\mathbf{g}(\mathbf{x} + \Delta\mathbf{x}) = \boldsymbol{\nabla} f(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{0} = \mathbf{g} + \mathbf{H}(\Delta\mathbf{x}) \quad \Rightarrow$$

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g} \quad \Rightarrow \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \left[\mathbf{H}\left(\mathbf{x}^k\right)\right]^{-1}\left[\mathbf{g}\left(\mathbf{x}^k\right)\right].$$

- Note that this is **exact for quadratic objective functions**, where $\mathbf{H} \equiv \mathbf{H}\left(\mathbf{x}^k\right) = \text{const}$.
- Also note that this is identical to using the Newton-Raphson method for solving the nonlinear system $\boldsymbol{\nabla}_{\mathbf{x}} f\left(\mathbf{x}^\star\right) = \mathbf{0}$.

Problems with Newton's Method

- Newton's method is exact for a quadratic function and converges in one step!
- For non-linear objective functions, however, Newton's method requires solving a linear system every step: **expensive**.
- It may not converge at all if the initial guess is not very good, or may converge to a saddle-point or maximum: **unreliable**.
- All of these are addressed by using variants of **quasi-Newton methods**:

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} - \alpha_k \mathbf{H}_k^{-1} \left[\mathbf{g}\left(\mathbf{x}^{k} \right) \right],$$

  where $0 < \alpha_k < 1$ and $\mathbf{H}_k$ is an approximation to the true Hessian.

General Formulation

- Consider the **constrained optimization problem**:

$$\begin{aligned} & \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) & \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = \mathbf{0} & \text{(equality constraints)} \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} & \text{(inequality constraints)} \end{aligned}$$

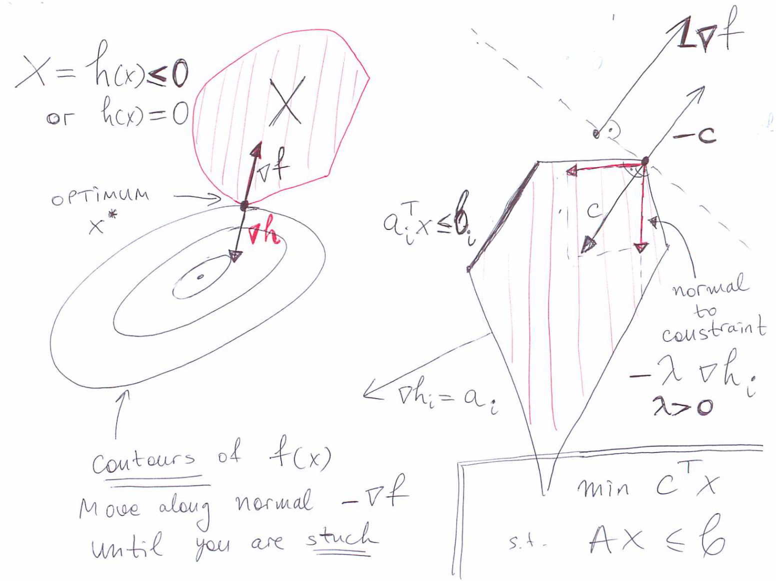
- Note that in principle only inequality constraints need to be considered since

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \quad \equiv \quad \begin{cases} \mathbf{h}(\mathbf{x}) \leq \mathbf{0} \\ \mathbf{h}(\mathbf{x}) \geq \mathbf{0} \end{cases}$$

  but this is not usually a good idea.

- We focus here on **non-degenerate cases** without considering various complications that may arise in practice.

Illustration of Lagrange Multipliers



Lagrange Multipliers: Single equality

- An equality constraint $h(\mathbf{x}) = 0$ corresponds to an $(n-1)-$dimensional **constraint surface** whose normal vector is $\boldsymbol{\nabla} h$.
- The illustration on previous slide shows that for a single smooth equality constraint, the gradient of the objective function must be parallel to the normal vector of the constraint surface:

$$\boldsymbol{\nabla} f \parallel \boldsymbol{\nabla} h \quad \Rightarrow \quad \exists \lambda \text{ s.t. } \boldsymbol{\nabla} f + \lambda \boldsymbol{\nabla} h = \mathbf{0},$$

  where λ is the **Lagrange multiplier** corresponding to the constraint $h(\mathbf{x}) = 0$.
- Note that the equation $\boldsymbol{\nabla} f + \lambda \boldsymbol{\nabla} h = \mathbf{0}$ is in **addition** to the constraint $h(\mathbf{x}) = 0$ itself.

Lagrange Multipliers: m equalities

- When m equalities are present,

$$h_1(\mathbf{x}) = h_2(\mathbf{x}) = \cdots = h_m(\mathbf{x})$$

the generalization is that the descent direction $-\nabla f$ must be in the span of the normal vectors of the constraints:

$$\nabla f + \sum_{i=1}^{m} \lambda_i \nabla h_i = \nabla f + (\nabla \mathbf{h})^T \boldsymbol{\lambda} = \mathbf{0}$$

where the **Jacobian** has the normal vectors as rows:

$$\nabla \mathbf{h} = \left\{ \frac{\partial h_i}{\partial x_j} \right\}_{ij}.$$

- This is a first-order **necessary optimality condition**.

Lagrange Multipliers: Single inequalities

- At the solution, a given inequality constraint $g_i(\mathbf{x}) \leq 0$ can be

$$\textbf{active} \text{ if } g_i(\mathbf{x}^\star) = 0$$
$$\textbf{inactive} \text{ if } g_i(\mathbf{x}^\star) < 0$$

- For inequalities, there is a definite sign (direction) for the constraint normal vectors:
  For an active constraint, you can move freely along $-\boldsymbol{\nabla} g$ but not along $+\boldsymbol{\nabla} g$.
- This means that for a single active constraint

$$\boldsymbol{\nabla} f = -\mu \boldsymbol{\nabla} g \quad \text{where} \quad \mu > 0.$$

Lagrange Multipliers: r inequalities

- The generalization is the same as for equalities

$$\nabla f + \sum_{i=1}^{r} \mu_i \nabla g_i = \nabla f + (\nabla \mathbf{g})^T \boldsymbol{\mu} = \mathbf{0},$$

but with the condition

$$\mu_i = 0 \text{ for inactive constraints.}$$
$$\mu_i > 0 \text{ for active constraints.}$$

- Putting equalities and inequalities together we get the **Karush-Kuhn-Tucker first-order necessary condition**: There exist Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $\boldsymbol{\mu} \in \mathbb{R}^r$ such that:

$$\nabla f + (\nabla \mathbf{h})^T \boldsymbol{\lambda} + (\nabla \mathbf{g})^T \boldsymbol{\mu} = \mathbf{0}, \quad \boldsymbol{\mu} \geq \mathbf{0} \text{ and } \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) = 0.$$

Lagrangian Function

$$\boldsymbol{\nabla} f + (\boldsymbol{\nabla}\mathbf{h})^T \boldsymbol{\lambda} + (\boldsymbol{\nabla}\mathbf{g})^T \boldsymbol{\mu} = \mathbf{0}$$

- We can rewrite this in the form of stationarity conditions

$$\boldsymbol{\nabla}_{\mathbf{x}}\mathcal{L} = \mathbf{0}$$

where $\mathcal{L}$ is the **Lagrangian function**:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}) + \sum_{i=1}^{r} \mu_i g_i(\mathbf{x})$$

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T [\mathbf{h}(\mathbf{x})] + \boldsymbol{\mu}^T [\mathbf{g}(\mathbf{x})]$$

Equality Constraints

- The **first-order necessary conditions** for equality-constrained problems are thus given by the stationarity conditions:

$$\begin{aligned} \nabla_{\mathbf{x}} \mathcal{L}\left(\mathbf{x}^{\star}, \boldsymbol{\lambda}^{\star}\right) &= \nabla f\left(\mathbf{x}^{\star}\right) + \left[\nabla \mathbf{h}(\mathbf{x}^{\star})\right]^{T} \boldsymbol{\lambda}^{\star} = \mathbf{0} \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(\mathbf{x}^{\star}, \boldsymbol{\lambda}^{\star}\right) &= \mathbf{h}(\mathbf{x}^{\star}) = \mathbf{0} \end{aligned}$$

- Note there are also **second order necessary and sufficient conditions** similar to unconstrained optimization.
- It is important to note that the solution $(\mathbf{x}^{\star}, \boldsymbol{\lambda}^{\star})$ is **not** a minimum or maximum of the Lagrangian (in fact, for convex problems it is a saddle-point, min in $\mathbf{x}$, max in $\boldsymbol{\lambda}$).
- Many numerical methods are based on Lagrange multipliers but we do not discuss it here.

Penalty Approach

- The idea is the convert the constrained optimization problem:

$$\begin{array}{ll} & \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ \text{s.t.} & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{array} .$$

  into an unconstrained optimization problem.
- Consider minimizing the **penalized function**

$$\mathcal{L}_\alpha(\mathbf{x}) = f(\mathbf{x}) + \alpha \left\|\mathbf{h}(\mathbf{x})\right\|_2^2 = f(\mathbf{x}) + \alpha \left[\mathbf{h}(\mathbf{x})\right]^T \left[\mathbf{h}(\mathbf{x})\right],$$

  where $\alpha > 0$ is a **penalty parameter**.
- Note that one can use **penalty functions** other than sum of squares.
- If the constraint is exactly satisfied, then $\mathcal{L}_\alpha(\mathbf{x}) = f(\mathbf{x})$.
  As $\alpha \to \infty$ violations of the constraint are penalized more and more, so that the equality will be satisfied with higher accuracy.

Penalty Method

- The above suggest the **penalty method** (see homework):
  For a monotonically diverging sequence $\alpha_1 < \alpha_2 < \cdots$, solve a **sequence of unconstrained problems**

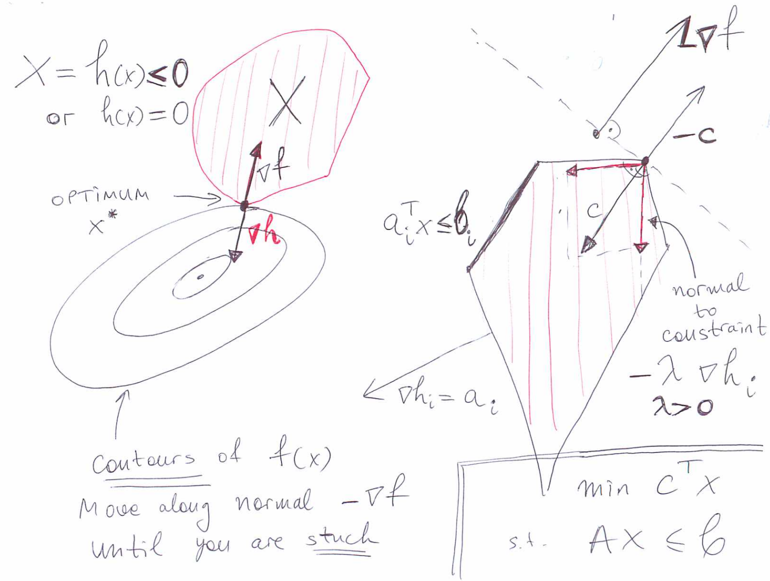
$$\mathbf{x}^k = \mathbf{x}(\alpha_k) = \arg\min_{\mathbf{x}} \left\{ \mathcal{L}_k(\mathbf{x}) = f(\mathbf{x}) + \alpha_k \left[\mathbf{h}(\mathbf{x})\right]^T \left[\mathbf{h}(\mathbf{x})\right] \right\}$$

  and the solution should converge to the optimum $\mathbf{x}^\star$,

$$\mathbf{x}^k \to \mathbf{x}^\star = \mathbf{x}(\alpha_k \to \infty).$$

- Note that one can use $\mathbf{x}^{k-1}$ as an initial guess for, for example, Newton's method.
- Also note that the problem becomes more and more ill-conditioned as α grows.
  A better approach uses Lagrange multipliers in addition to penalty (**augmented Lagrangian**).

Illustration of Constrained Programming



Linear Programming

- Consider **linear programming** (see illustration)

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left\{\mathbf{c}^T\mathbf{x}\right\}$$
$$\text{s.t.}\qquad \mathbf{A}\mathbf{x}\leq\mathbf{b}\qquad .$$

- The feasible set here is a **polytope** (polygon, polyhedron) in $\mathbb{R}^n$, consider for now the case when it is bounded, meaning there are at least $n+1$ constraints.
- The optimal point is a **vertex** of the polyhedron, meaning a point where (generically) n constraints are **active**,

$$\mathbf{A}_{act}\mathbf{x}^{\star}=\mathbf{b}_{act}.$$

- Solving the problem therefore means finding the subset of active constraints:

  Combinatorial search problem, solved using the **simplex algorithm** (search along the edges of the polytope).
- Lately **interior-point methods** have become increasingly popular (move inside the polytope).

Conclusions/Summary

- Optimization, or **mathematical programming**, is one of the most important numerical problems in practice.
- Optimization problems can be **constrained** or **unconstrained**, and the nature (linear, convex, quadratic, algebraic, etc.) of the functions involved matters.
- Finding a **global minimum** of a general function is virtually **impossible** in high dimensions, but very important in practice.
- An unconstrained local minimum can be found using **direct search**, **gradient descent**, or **Newton-like methods**.
- Equality-constrained optimization is **tractable**, but the best method **depends on the specifics**.
  We looked at penalty methods only as an illustration, not because they are good in practice!
- Constrained optimization is tractable for the convex case, otherwise often hard, and even **NP-complete** for **integer programming**.