Conclusion and Summary

Optimization Techniques (ENGG*6140)

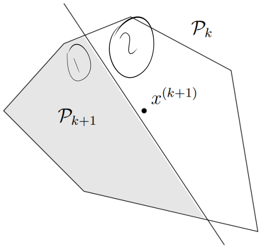
School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Winter 2023

Additional Notes

Cutting-Plane Methods

- **Cutting plane** methods, also called the **localization** methods, are a family of methods which start with a large feasible set containing the solution to be found.
- Then, iteratively they reduce the feasible set by cutting off some piece of it [1].
- For example, a cutting-plane method starts with a polyhedron feasible set. It finds a plane at every iteration which divides the feasible sets into two disjoint parts one of which contains the ultimate solution. It gets rid of the part without solution and reduces the volume of the polyhedron. This is repeated until the polyhedron feasible set becomes very small and converges to the solution.



credit of image: [1]

Cutting-Plane Methods

- This is somewhat a generalization of the **bisection method**, also called the **binary search method**, which was used for root-finding [2] but later it was used for convex optimization. The bisection method halves the feasible set and removes the part without the solution, at every iteration

```
Input: l and u
for iteration k = 0, 1, . . . do
    x^(k+1) := (l+u)/2
    if ∇f(x) < 0 then
        l := x
    else
        u := x
    Check the convergence criterion
    if converged then
        return x^(k+1)
```

Algorithm : The bisection algorithm

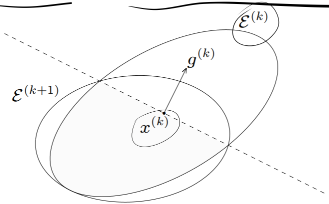
- Some of the important cutting-plane methods are **center of gravity method**, **Maximum Volume Ellipsoid (MVE) cutting-plane method**, **Chebyshev center cutting-plane method**, and **Analytic Center Cutting-Plane Method (ACCPM)** [3, 4, 5]. Similar to subgradient methods, cutting-plane methods can be used for optimizing non-smooth functions.

Ellipsoid Method

- **Ellipsoid method** was developed by several people [6, 7].
- It was proposed in 1976 and 1977 [8, 9, 10, 11].
- It was initially applied to liner programming in a famous paper in 1979 [12].
- It is similar to cutting-plane methods in cutting some part of feasible set iteratively.
- At every iteration, it finds an ellipsoid centered at the current solution:

$$\mathcal{E}(\boldsymbol{x}^{(k)}, \boldsymbol{P}) := \{\boldsymbol{z} \,|\, (\boldsymbol{z} - \boldsymbol{x}^{(k)})^\top \boldsymbol{P}^{-1}(\boldsymbol{z} - \boldsymbol{x}^{(k)}) \leq 1\},$$

  where $\boldsymbol{P} \in \mathbb{S}^d_{++}$ (is symmetric positive definite).
- It removes half of the ellipsoid which does not contain the solution.
- Again, another ellipsoid is found at the updated solution. This is repeated until the ellipsoid of iteration is very small and converges to the solution.



credit of image: [13]

Minimax and Maximin Problems

- Consider a function of two variables, $f(\boldsymbol{x}_1, \boldsymbol{x}_2)$, and the following optimization problem:

$$\underset{\boldsymbol{x}_1}{\text{minimize}} \Big(\underset{\boldsymbol{x}_2}{\text{maximize}}\ f(\boldsymbol{x}_1, \boldsymbol{x}_2)\Big). \tag{1}$$

  In this problem, we want to minimize the function w.r.t. one of the variables and maximize it w.r.t the other variable. This optimization problem is called the **minimax problem**.
- We can change the order of this problem to have the so-called **maximin problem**:

$$\underset{\boldsymbol{x}_1}{\text{maximize}} \Big(\underset{\boldsymbol{x}_2}{\text{minimize}}\ f(\boldsymbol{x}_1, \boldsymbol{x}_2)\Big). \tag{2}$$

- Under certain conditions [14], the minimax and maximin problems are equivalent if the variables of maximization (or minimization) stay the same. In other words, under some conditions, we have [14]:

$$\underset{\boldsymbol{x}_1}{\text{minimize}} (\underset{\boldsymbol{x}_2}{\text{maximize}}\, f(\boldsymbol{x}_1, \boldsymbol{x}_2)) = \underset{\boldsymbol{x}_2}{\text{maximize}} (\underset{\boldsymbol{x}_1}{\text{minimize}}\, f(\boldsymbol{x}_1, \boldsymbol{x}_2)).$$

- In the minimax and maximin problems, the two variables have conflicting or contrastive desires; one of them wants to maximize the function while the other wants to minimize it. Hence, they are widely used in the field of **game theory** as important strategies [15].

Summary

Summary of what we learned in this course

- The course provided **the main methods of optimization**.
- We started with **preliminaries** including sets, norms, functions, local/global minimizer, derivatives, gradient, Jacobian, Hessian, convexity of sets, and convexity of functions.
- We introduced the **standard problems** (e.g., convex problem, linear programming, quadratic programming, semidefinite programming, etc).
- We covered **linear programming** (the **simplex algorithm**) and **integer linear programming** for continuous and discrete linear problems, respectively.
- We introduced the **Karush-Kuhn-Tucker (KKT) conditions** along with the **Lagrangian function** and the **method of Lagrange multipliers**.
- We covered **unconstrained and constrained first-order optimization** which are gradient methods and include **gradient descent, backpropagation, AGM, SGD, SAG, Adam, neural network, and proximal methods**.
- The **unconstrained and constrained second order optimization** techniques, including the **Newton's method** and **interior-point method**, were covered in order to be able to solve all convex optimization problems (this method also works fairly well on nonconvex problems). We also covered **decomposition methods** and **conjugate gradient** for accelerating Newton's method as well as the **quasi-Newton**'s methods.

Summary of what we learned in this course

- We also went through **distributed optimization** (such as **alternating optimization** and **ADMM**) in order to solve complex multivariate optimization problems.
- We covered **non-smooth optimization** including **approximation by convex conjugate and Huber function**, **proximal algorithm** and **soft thresholding**, **coordinate descent**, and **subgradients**.
- We covered **non-convex optimization** including local optimization method (**sequential convex programming**) and global optimization method (**branch and bound**).
- We covered important **search-based (metaheuristic) optimization** such as **genetic algorithm**, **particle swarm optimization**, **simulated annealing**, and the **Nelder-Mead simplex algorithm**.
- Finally, we also introduced **Riemannian optimization** for optimization on the Riemannian matrix manifolds.
- Additional methods which we mentioned: **cutting-plane** methods, **ellipsoid** method, **minimax** and **maximin** problems

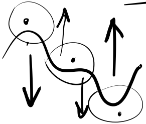
Dealing with an Optimization Problem

Dealing with an Optimization Problem

- Assume we have an optimization problem where we want to **minimize or maximize** a cost function and we might have several **equality** and/or **inequality constraints** as the feasibility set.
- If the problem is **unconstrained**:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}).$$

  - take derivative and set it to zero (first-order optimality condition) (high school rule):



$$\nabla f(\boldsymbol{x}) \overset{\text{set}}{=} \mathbf{0} \implies \boldsymbol{x}^* = \text{expression without } \boldsymbol{x}, \quad (\text{closed-form}),$$

$$\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}, \quad \forall \boldsymbol{x} \text{ s.t. } \|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \epsilon.$$

  - if not-closed-form solution, we should use numerical optimization:
    - ★ first-order optimization: needs first-order derivative $\nabla f(\boldsymbol{x})$
    - ★ second-order optimization: needs first-order derivative $\nabla f(\boldsymbol{x})$ and second-order derivative $\nabla^2 f(\boldsymbol{x})$

Dealing with an Optimization Problem

- If the problem is **constrained**:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & y_i(\boldsymbol{x}) \leq 0, \; i \in \{1, \dots, m_1\}, \\ & h_i(\boldsymbol{x}) = 0, \; i \in \{1, \dots, m_2\}. \end{aligned}$$

We can embed the constraint into the objective function using **penalty** (**regularization**) term:

- **soft penalty**:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) + \sum_{i=1}^{m_1} \lambda_i y_i(\boldsymbol{x}) + \sum_{i=1}^{m_2} \nu_i h_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \eta \Omega(\boldsymbol{x} \in \mathcal{S}),$$

where:

$$\Omega(\boldsymbol{x} \in \mathcal{S}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{S} \\ \text{increasing function w.r.t. not being in feasibility set} & \text{if } \boldsymbol{x} \notin \mathcal{S} \end{cases}$$

Dealing with an Optimization Problem

- If the problem is **constrained**:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & y_i(\boldsymbol{x}) \leq 0, \; i \in \{1, \dots, m_1\}, \\ & h_i(\boldsymbol{x}) = 0, \; i \in \{1, \dots, m_2\}. \end{aligned}$$

We can embed the constraint into the objective function using **penalty** (**regularization**) term:

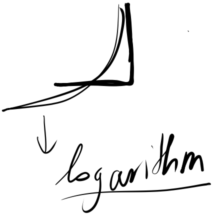
- **hard penalty**:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) + \eta \mathbb{I}(\boldsymbol{x} \in \mathcal{S}),$$

where:

$$\mathbb{I}(\boldsymbol{x} \in \mathcal{S}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{S} \\ \infty & \text{if } \boldsymbol{x} \notin \mathcal{S} \end{cases}$$

- approximation of hard penalty: such as log-barrier method.



Dealing with an Optimization Problem

- If the problem is **linear programming** (affine cost and constraints), we can solve it using the **simplex algorithm**; any of its implementations such as the **tableau method**.
- If the problem is **integer linear programming** (affine cost and constraints and some or all variables are integer), we should use the **branch and bound** method. In each subproblem of the tree, we use the **simplex** algorithm; any of its implementations such as the **tableau method**.
- If the optimization problem is not linear but not very complicated, we can solve it using the **method of Lagrange multipliers** which makes use of the **KKT conditions**. If it is complicated, solving the system of equations in the method of Lagrange multipliers is very hard.

Dealing with an Optimization Problem

- If the optimization problem is **unconstrained**:
  - slower but easier: we can use unconstrained first-order optimization methods, such as **gradient descent, backpropagation, AGM, SGD, SAG, Adam, neural network, and unconstrained proximal methods**.
  - faster but harder: we can use unconstrained second-order optimization method, such as **Newton's method**.
- If the optimization problem is **constrained**:
  - slower but easier: we can use constrained first-order optimization methods, such as **projected gradient method, and constrained proximal methods**.
  - faster but harder: we can use constrained second-order optimization method, such as **interior-point method** and **log-barrier method**.
- We can use **decomposition methods**, **conjugate gradient**, or **quasi-Newton's method** to make second-order optimization easier.

Dealing with an Optimization Problem

- If the optimization problem is **distributed** or with **multiple variables**, we can use distributed optimization such as **alternating optimization** and **ADMM**. We can use multiple servers or cores to make it **parallel**.
- If the cost in the optimization problem is **non-smooth** (non-differentiable) at least at one of the points in its domain/feasibility set, we can use **non-smooth optimization** including **approximation by convex conjugate and Huber function**, **proximal algorithm** and **soft thresholding**, **coordinate descent**, and **subgradients**.
- If the optimization problem is **non-convex** (i.e., cost is not a convex function and/or the feasibility set is non-convex set), we can use **non-convex optimization**
  - fast but not guarantee to find the global solution: local optimization method (**sequential convex programming**)
  - slow but guarantees to find the global solution: global optimization method (**branch and bound**)

Dealing with an Optimization Problem

- We use either **non-convex optimization** or **search-based** optimization (**metaheuristic** optimization) if:
  - if the optimization problem has a **complicated** (highly con-convex) optimization landscape,
  - or when the **gradient** of function is **hard to compute**,
  - or when the function is not known but it works as a **black-box**, i.e., it outputs a value for each input fed to it,

  Some examples for **search-based (metaheuristic) optimization** are **genetic algorithm**, **particle swarm optimization**, **simulated annealing**, and the **Nelder-Mead simplex algorithm**.

- If the feasibility set (set of constraints) can be seen as a **Riemannian manifold**, we can use **Riemannian optimization**. Some examples are orthogonal matrix (Stiefel manifold), projection matrix onto subspace (Grassmannian manifold), and Symmetric Positive Definite (SPD) matrices (SPD manifold).

$x \rightarrow f \rightarrow f(x)$

$$\min_X \; tr(X^T A X) \quad s.t. \; X^T X = I \rightarrow \text{Stiefel}$$

$$\rightarrow \min_X \; tr(X^T A X) \quad s.t. \; X \in \mathcal{M}_{\text{Stiefel}}$$

KKT Backbone!

- I think now that you know the theory behind many of the optimization algorithms, you can see the **Nirvana (enlightenment) moment of optimization**:
- **The backbone of most of optimization is KKT conditions and Lagrangian!**

Acknowledgement

- Some slides of this slide deck are inspired by the lectures of Prof. Stephen Boyd at the Stanford University.
- Our tutorial also has some of the materials of this slide deck: [16]

References

[1] S. Boyd and L. Vandenberghe, "Localization and cutting-plane methods," tech. rep., Stanford EE 364b lecture notes, 2007.

[2] R. L. Burden and J. D. Faires, *Numerical Analysis*.
PWS Publishers, 1963.

[3] J.-L. Goffin and J.-P. Vial, "On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm," *Mathematical Programming*, vol. 60, no. 1, pp. 81–92, 1993.

[4] Y. Nesterov, "Cutting plane algorithms from analytic centers: efficiency estimates," *Mathematical Programming*, vol. 69, no. 1, pp. 149–176, 1995.

[5] D. S. Atkinson and P. M. Vaidya, "A cutting plane algorithm for convex programming that uses analytic centers," *Mathematical Programming*, vol. 69, no. 1, pp. 1–43, 1995.

[6] P. Wolfe, "Invited noteâsome references for the ellipsoid algorithm," *Management Science*, vol. 26, no. 8, pp. 747–749, 1980.

[7] S. Rebennack, "Ellipsoid method," *Encyclopedia of Optimization*, pp. 890–899, 2009.

[8] N. Z. Shor, "Cut-off method with space extension in convex programming problems," *Cybernetics*, vol. 13, no. 1, pp. 94–96, 1977.

References (cont.)

[9] D. Yudin and A. S. Nemirovski, "Informational complexity and efficient methods for the solution of convex extremal problems," *Èkon Math Metod, English translation: Matekon*, vol. 13, no. 2, pp. 22–45, 1976.

[10] D. Yudin and A. Nemirovski, "Evaluation of the informational complexity of mathematical programming problems," *Èkon Math Metod, English translation: Matekon*, vol. 13, no. 2, pp. 3–24, 1977.

[11] D. Yudin and A. Nemirovski, "Optimization methods adapting to the "significant" dimension of the problem," *Autom Telemekhanika, English translation: Automation and Remote Control*, vol. 38, no. 4, pp. 513–524, 1977.

[12] L. G. Khachiyan, "A polynomial algorithm in linear programming," in *Doklady Akademii Nauk*, vol. 244, pp. 1093–1096, Russian Academy of Sciences, 1979.

[13] S. Boyd and C. Barratt, "Ellipsoid method," *Notes for EE364B, Stanford University*, vol. 2008, 2008.

[14] D.-Z. Du and P. M. Pardalos, *Minimax and applications*, vol. 4. Springer Science & Business Media, 2013.

[15] R. J. Aumann and M. Maschler, "Some thoughts on the minimax principle," *Management Science*, vol. 18, no. 5-part-2, pp. 54–63, 1972.

References (cont.)

[16] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "KKT conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey," *arXiv preprint arXiv:2110.01858*, 2021.