Non-convex Optimization

Optimization Techniques (ENGG*6140)

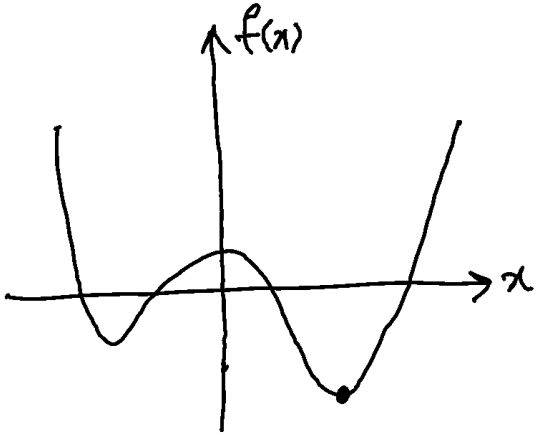
School of Engineering,
University of Guelph, ON, Canada

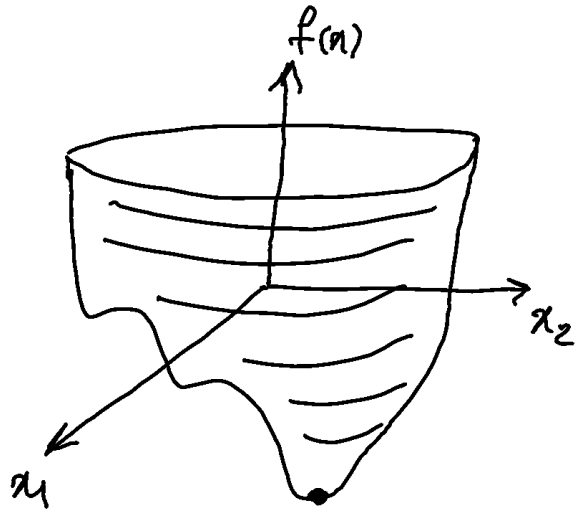
Course Instructor: Benyamin Ghojogh
Winter 2023

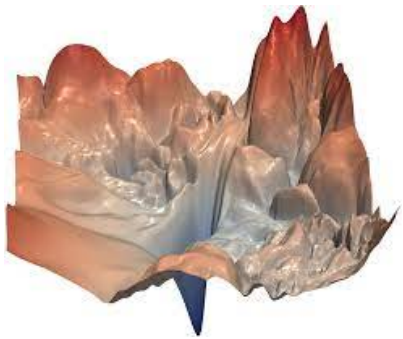
Non-convex Function and Optimization

Non-convex function

- We might have non-convex cost functions (recall the preliminaries) (credit of second image: [1]):







- In these cases, we need non-convex optimization.

Non-convex optimization

- Consider the following optimization problem:

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

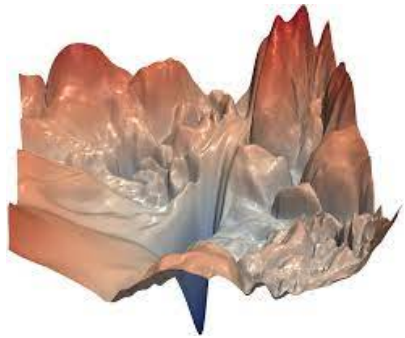
  where the functions $f(.)$, $y_i(.)$, and $h_i(.)$ are not necessarily convex.
- The introduced optimization methods can also work for non-convex problems but they do not guarantee to find the global optimum.
- They can find local minimizers which depend on the random initial solution.
- For example, the optimization landscape of **neural network** is highly nonlinear and non-convex but backpropagation works very well for it.

Non-convex optimization in neural networks

- The optimization landscape of **neural network** is highly nonlinear and non-convex but backpropagation works very well for it.
  - **Question:** Assume the loss function of neural network is the mean squared error:

$$\frac{1}{b}\sum_{i=1}^{b}\|\boldsymbol{y}_i - \boldsymbol{l}_i\|_2^2,$$

    where b is the mini-batch size, $\boldsymbol{y}_i$ is the output of network, and $\boldsymbol{l}_i$ is the one-hot encoded label for the i-th data in the mini-batch.
  - This loss is clearly quadratic and convex. Then, why is the neural network highly non-convex? (credit of image: [1])



- The reason for this is explained in this way: every layer of neural network pulls data to the feature space such as in kernels [2]. In the high-dimensional feature space, all local minimizers are almost global minimizers because the local minimum values are almost equal in that space [3]. Also see [4, 5, 6] to understand why backpropagation optimization works well even in highly non-convex optimization.

Non-convex optimization

- As was explained, the already introduced first-order and second-order optimization methods can work fairly well for non-convex problems by finding local minimizers depending on the initial solution.
- However, there exist some specific methods for **non-convex programming**, divided into two categories.
  - The **local optimization methods** are faster but do not guarantee to find the global minimizer.
  - The **global optimization methods** find the global minimizer but are usually slow to find the answer [7].
- Example for local optimization methods:
  - **Sequential Convex Programming (SCP)** [8] is an example for local optimization methods.
  - It is based on a sequence of convex approximations of the non-convex problem.
  - It is closely related to **Sequential Quadratic Programming (SQP)** [9] which is used for constrained nonlinear optimization.
- Example for global optimization methods:
  - **Branch and bound method**, first proposed in 1960 [10], is an example for the global optimization methods.
  - It divides the optimization landscape, i.e. the feasible set, into local parts by a binary tree and solves optimization in every part.
  - It checks whether the solution of a part is the global solution or not.
- In this slide deck, we explain SCP which is a faster but local method.
- Note that another approach for highly non-convex optimization is **metaheuristic (search-based) optimization** which will be briefly introduced later.

Convex Approximation

Convex Approximation

- Recall Eq. (1):

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

- SCP iteratively solves a convex problem where, at every iteration, it approximates the non-convex problem (1) with a convex problem, based on the current solution, and restricts the variable to be in a so-called **trust region** [11].
- The **trust region** makes sure that the variable stays in a locally convex region of the optimization problem.
- At the iteration k of SCP, we solve the following convex problem:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \widehat{f}(\boldsymbol{x}) \\ \text{subject to} \quad & \widehat{y}_i(\boldsymbol{x}) \leq 0,\ i \in \{1, \ldots, m_1\}, \\ & \widehat{h}_i(\boldsymbol{x}) = 0,\ i \in \{1, \ldots, m_2\}, \\ & \boldsymbol{x} \in \mathcal{T}^{(k)}, \end{aligned} \tag{2}$$

  where $\widehat{f}(.)$, $\widehat{y}_i(.)$, and $\widehat{h}_i(.)$, are convex approximations of functions $f(.)$, $y_i(.)$, and $h_i(.)$, and $\mathcal{T}^{(k)}$ is the trust region at iteration k.
- This approximated convex problem is also solved iteratively itself using one of the previously introduced methods such as the interior-point method.
- There exist several approaches for convex approximation of the functions. In the following, we introduce some of these approaches.

Convex Approximation by Taylor Series Expansion

- The non-convex functions $f(.)$, $y_i(.)$, and $h_i(.)$ can be approximated by **affine functions** (i.e., **first-order Taylor series expansion**) to become convex. For example, the function $f(.)$ is approximated as:

$$\widehat{f}(\boldsymbol{x}) = f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(k)}). \tag{3}$$

- The functions can also be approximated by **quadratic functions** (i.e., **second-order Taylor series expansion**) to become convex. For example, the function $f(.)$ is approximated as:

$$\widehat{f}(\boldsymbol{x}) = f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(k)})^\top \boldsymbol{P}(\boldsymbol{x} - \boldsymbol{x}^{(k)}), \tag{4}$$

  where $\boldsymbol{P} = \Pi_{\mathbb{S}_+^d}(\nabla^2 f(\boldsymbol{x}^{(k)}))$ is projection of Hessian onto the symmetric positive semi-definite cone. This projection is performed by setting the negative eigenvalues of Hessian to zero.

- The same approaches can be used for approximation of functions $y_i(.)$ and $h_i(.)$ using first- or second-order Taylor expansion.

Convex Approximation by Particle Method

- We can approximate the functions $f(.)$, $y_i(.)$, and $h_i(.)$ in the domain of trust region using **regression**. This approach is named the **particle method** [7].
- Let $\{\boldsymbol{x}_i \in \mathcal{T}^{(k)}\}_{i=1}^{m}$ be m points which lie in the trust region. We can use **least-squares quadratic regression** to make the functions convex in the trust region:

$$\begin{aligned} \underset{a\in\mathbb{R},\boldsymbol{b}\in\mathbb{R}^d,\boldsymbol{P}\in\mathbb{S}^d_{++}}{\text{minimize}} \quad & \sum_{i=1}^{m}\Big(\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})^\top \boldsymbol{P}(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})+\boldsymbol{b}^\top(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})+a-f(\boldsymbol{x}_i)\Big)^2 \\ \text{subject to} \quad & \boldsymbol{P}\succeq \boldsymbol{0}. \end{aligned} \tag{5}$$

- Then, the function $f(.)$ is replaced by its convex approximation:

$$\widehat{f}(\boldsymbol{x}) = (1/2)(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})^\top \boldsymbol{P}(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})+\boldsymbol{b}^\top(\boldsymbol{x}_i-\boldsymbol{x}^{(k)})+a.$$

- The same approach can be used for approximation of functions $y_i(.)$ and $h_i(.)$.

Convex Approximation by Quasi-linearization

- Another approach for convex approximation of functions $f(.)$, $y_i(.)$, and $h_i(.)$ is **quasi-linearization**.
- We should state the function $f(.)$ in the form $f(\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{x})\,\boldsymbol{x} + c(\boldsymbol{x})$.
- For example, we can use the second-order Taylor series expansion to do this:

$$f(\boldsymbol{x}) \approx \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + a = (\frac{1}{2}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{b})^\top \boldsymbol{x} + a,$$

  so we use $\boldsymbol{A}(\boldsymbol{x}) := ((1/2)\boldsymbol{P}\boldsymbol{x} + \boldsymbol{b})^\top$ and $c(\boldsymbol{x}) := a$ which depend on the Taylor expansion of $f(\boldsymbol{x})$.
- Hence, the convex approximation of function $f(.)$ can be:

$$\widehat{f}(\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{x}^{(k)})\,\boldsymbol{x} + c(\boldsymbol{x}^{(k)}) = (\frac{1}{2}\boldsymbol{P}\boldsymbol{x}^{(k)} + \boldsymbol{b})^\top \boldsymbol{x} + a. \tag{6}$$

- The same approach can be used for approximation of functions $y_i(.)$ and $h_i(.)$.

Trust Region

Formulation of Trust Region

- The trust region can be a **box** around the point at that iteration:

$$\mathcal{T}^{(k)} := \{\boldsymbol{x} \,|\, |x_j - x_j^{(k)}| \leq \rho_i, \forall j \in \{1, \ldots, d\}\}. \tag{7}$$

  where x_j and $x_j^{(k)}$ are the j-th element of $\boldsymbol{x}$ and $\boldsymbol{x}^{(k)}$, respectively, and ρ_i is the bound of box for the j-th dimension.

- Another option for trust region is an **ellipse** around the point to have a **quadratic** trust region:

$$\mathcal{T}^{(k)} := \{\boldsymbol{x} \,|\, (\boldsymbol{x} - \boldsymbol{x}^{(k)})^\top \boldsymbol{P}^{-1} (\boldsymbol{x} - \boldsymbol{x}^{(k)}) \leq \rho\}, \tag{8}$$

  where $\boldsymbol{P} \in \mathbb{S}^d_{++}$ (is symmetric positive definite) and $\rho > 0$ is the radius of ellipse.

Updating Trust Region

- The trust region gets updated in every iteration of SCP. In the following, we explain how the trust region can be updated.
- Recall Eq. (1):

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

- First, we embed the constraints in the objective function of problem (1):

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \phi(\boldsymbol{x}) := f(\boldsymbol{x}) + \lambda \Big(\sum_{i=1}^{m_1} \big(\max(y_i(\boldsymbol{x}), 0) \big)^2 + \sum_{i=1}^{m_1} |h_i(\boldsymbol{x})|^2 \Big), \tag{9}$$

  where $\lambda > 0$ is the regularization parameter. This is called the **exact penalty method** (1994) [12] because it penalizes violation from the constraints.
- For large enough regularization parameter (which gives importance to violation of constraints), the solution of problem (9) is exactly equal to the solution of problem (1). That is the reason for the term "exact" in the name "exact penalty method".

Updating Trust Region

- Recall Eq. (2):

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \widehat{f}(\boldsymbol{x}) \\ \text{subject to} \quad & \widehat{y}_i(\boldsymbol{x}) \leq 0,\ i \in \{1,\dots,m_1\}, \\ & \widehat{h}_i(\boldsymbol{x}) = 0,\ i \in \{1,\dots,m_2\}, \\ & \boldsymbol{x} \in \mathcal{T}^{(k)}, \end{aligned}$$

- We found Eq. (9):

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \phi(\boldsymbol{x}) := f(\boldsymbol{x}) + \lambda\Big(\sum_{i=1}^{m_1}\big(\max(y_i(\boldsymbol{x}),0)\big)^2 + \sum_{i=1}^{m_1}|h_i(\boldsymbol{x})|^2\Big).$$

- Similar to Eq. (9), we define:

$$\widehat{\phi}(\boldsymbol{x}) := \widehat{f}(\boldsymbol{x}) + \lambda\Big(\sum_{i=1}^{m_1}\big(\max(\widehat{y}_i(\boldsymbol{x}),0)\big)^2 + \sum_{i=1}^{m_1}|\widehat{h}_i(\boldsymbol{x})|^2\Big), \tag{10}$$

for the problem (2). At the iteration k of SCP, let $\widehat{x}^{(k)}$ be the solution of the convex approximated problem (2) using any method such as the interior-point method.

- We calculate the predicted and exact decreases which are $\widehat{\delta} := \phi(\boldsymbol{x}^{(k)}) - \widehat{\phi}(\widehat{\boldsymbol{x}})$ and $\delta := \phi(\boldsymbol{x}^{(k)}) - \phi(\widehat{\boldsymbol{x}})$, respectively.

Updating Trust Region

- We calculate the predicted and exact decreases which are $\widehat{\delta} := \phi(\boldsymbol{x}^{(k)}) - \widehat{\phi}(\widehat{\boldsymbol{x}})$ and $\delta := \phi(\boldsymbol{x}^{(k)}) - \phi(\widehat{\boldsymbol{x}})$, respectively.
- Two cases may happen:
  - We have progress in optimization if $\alpha\widehat{\delta} \leq \delta$ where $0 < \alpha < 1$ (e.g., $\alpha = 0.1$). In this case, we accept the approximate solution:

    $$\boldsymbol{x}^{(k+1)} := \widehat{\boldsymbol{x}},$$

    and we increase the size of trust region, for the next iteration of SCP, by:

    $$\rho^{(k+1)} := \beta\rho^{(k)},$$

    where $\beta \geq 1$ (e.g., $\beta = 1.1$).
  - We do not have progress in optimization if $\alpha\widehat{\delta} > \delta$. In this case, we reject the approximate solution:

    $$\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)},$$

    and we decrease the size of trust region, for the next iteration of SCP, by:

    $$\rho^{(k+1)} := \gamma\rho^{(k)},$$

    where $0 < \gamma < 1$ (e.g., $\gamma = 0.5$).
- In summary, the trust region is expanded if we find a good solution; otherwise, it is made smaller.

Branch and Bound Method

Branch and Bound Method

- Branch and bound method, proposed in 1960 [10], is a **global** optimization method, which is slow but find the global optimum.
- The idea of this method:
  - divide the feasibility set region into two convex sets (e.g., rectangular (hyper-cubic) regions).
  - in each region, find an upper bound and a lower bound.
  - choose the better region among these two.
  - do it iteratively until convergence.
- Approximation of **lower bound** of optimal value f^* in a region:
  - it can be the solution of the **convex relaxation** of the optimization problem.
- Approximation of **upper bound** of optimal value f^* in a region:
  - it can be the function value at **any point in the region**.
- The more we progress in the algorithm, the closer (tighter) the lower bound and the upper bound get to each other.

Branch and Bound Method

The algorithm of branch and bound method:

1. start with the feasibility set as the region.
2. calculate the lower bound l and upper bound u of the feasibility set. Set the overall lower bound and overall upper bound to the lower bound and upper bound, respectively:

$$l_{\text{overall}} = l, \quad u_{\text{overall}} = u$$

3. loop over iterations:
   1. split the region into left and right regions.
   2. calculate the lower bound and upper bound of the left and right regions, $l_{\text{left}}, u_{\text{left}}, l_{\text{right}}, u_{\text{right}}$.
   3. Update the overall lower bound if the largest lower bound among left and right regions is larger than the overall lower bound.

$$l_{\text{overall}} = \max(l_{\text{left}}, l_{\text{right}}, l_{\text{overall}}).$$

   4. Update the overall upper bound if the smallest upper bound among left and right regions is smaller than the overall upper bound.

$$u_{\text{overall}} = \min(u_{\text{left}}, u_{\text{right}}, u_{\text{overall}}).$$

   5. if $u_{\text{overall}} - l_{\text{overall}} < \epsilon$: terminate and, for minimization, return l_{left} and its corresponding $\boldsymbol{x}^*$ if $l_{\text{left}} < l_{\text{right}}$. But return l_{right} and its corresponding $\boldsymbol{x}^*$ if $l_{\text{right}} < l_{\text{left}}$.

Acknowledgement

- Some slides of this slide deck are inspired by the lectures of Prof. Stephen Boyd at the Stanford University.
- Our tutorial also has the materials of this slide deck: [13]

References

[1] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[2] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Reproducing kernel Hilbert space, Mercer's theorem, eigenfunctions, Nyström method, and use of kernels in machine learning: Tutorial and survey," *arXiv preprint arXiv:2106.08443*, 2021.

[3] S. Feizi, H. Javadi, J. Zhang, and D. Tse, "Porcupine neural networks:(almost) all local optima are global," *arXiv preprint arXiv:1710.02196*, 2017.

[4] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 742–769, 2018.

[5] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," *Advances in neural information processing systems*, 2019.

[6] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*, pp. 242–252, 2019.

[7] J. Duchi, S. Boyd, and J. Mattingley, "Sequential convex programming," tech. rep., Notes for EE364b, Stanford University, 2018.

References (cont.)

[8] Q. T. Dinh and M. Diehl, "Local convergence of sequential convex programming for nonconvex optimization," in *Recent Advances in Optimization and its Applications in Engineering*, pp. 93–102, Springer, 2010.

[9] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[10] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[11] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[12] G. Di Pillo, "Exact penalty methods," in *Algorithms for Continuous Optimization*, pp. 209–253, Springer, 1994.

[13] 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.