CUB Optimization Methods in Machine Learning, Spring 2024

Test Exam

REFERENCE SOLUTION

Exam total time is 60 minutes. During exam no materials can be used. For each task you may get 1 point. The exam grade is computed as a sum of points for all tasks divided by 10.

1. The function $f(\boldsymbol{x})$ belongs to the class with Lipschitz Hessian, if the function is at least two times continuously differentiable and its Hessian satisfies the following Lipschitz condition:

$$\|\nabla^2 f(\boldsymbol{x}) - \nabla^2 f(\boldsymbol{y})\| \leq M\|\boldsymbol{x} - \boldsymbol{y}\| \ \forall \ \boldsymbol{x}, \boldsymbol{y}$$

Here M is some constant.

The three-times continuously differentiable one-dimensional function would belong to this class if and only if its third derivative is globally bounded. So the function $f(x) = x^2$ belongs to this class, but the function $f(x) = \exp(x)$ does not belong to the class.

2. Let's consider the following optimization problem:

$$f(\boldsymbol{x}) \to \min_{\boldsymbol{x}},$$

where f is a smooth function.

(Necessary condition) If $\boldsymbol{x}_*$ is a local minima of f, then $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_*) = 0$, $\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}_*)$ is a non-negatively defined matrix.

(Sufficient condition) If $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_*) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}_*)$ is strictly positively defined matrix, then $\boldsymbol{x}_*$ is a local minima of f.

3. Let's consider a sequence of strictly positive numbers r_k and the following limits: $\alpha = \overline{\lim}_{k\to+\infty} \frac{r_{k+1}}{r_k}$, $\beta = \underline{\lim}_{k\to+\infty} \frac{r_{k+1}}{r_k}$. Then

- If $\alpha = 0$, then the sequence has a superlinear rate;
- If $0 < \alpha < 1$, then the sequence has a linear rate;
- If $\beta = 1$, then the sequence has a sublinear rate;
- If $\beta < 1$ and $\alpha \geq 1$ then nothing can be said.

Let's consider the following sequence:

$$r_k = \begin{cases} \frac{1}{2^k}, \text{if k is odd} \\ r_{k-1}, \text{ if k is even} \end{cases}$$

The sequence has a linear convergence rate since it can be upper bounded with a sequence $1/2^{k-1}$. However, $\alpha = 1$, $\beta = 1/4$, so nothing can be said from the test of ratios.

4. (Intersection of convex sets) $Q = \cap_i Q_i$. If all Q_i are convex, then Q is convex.

(Affine image) Let's consider some affine transformation $A(\boldsymbol{x}) = L(\boldsymbol{x}) + \boldsymbol{b}$. Here L is some linear operator. Then $A(Q) = \{\boldsymbol{y} : \exists \boldsymbol{x} \in Q : \boldsymbol{y} = A(\boldsymbol{x})\}$ is convex if Q is convex.

(Affine preimage) Let's consider some affine transformation $A(\boldsymbol{x}) = L(\boldsymbol{x}) + \boldsymbol{b}$. Here L is some linear operator. Then $A^{-1}(Q) = \{\boldsymbol{x} : A(\boldsymbol{x}) \in Q\}$ is convex if Q is convex.

(Cartesian product of convex sets) $Q = Q_1 \times Q_2 \times \cdots \times Q_n$ is convex if all Q_i are convex.

5. Suppose $\boldsymbol{x} \in \mathbb{R}^n$. GD one iteration complexity is $O(n)$. Newton method requires solving a linear system, so its computation complexity is $O(n^3)$. Conjugate gradient has iteration complexity $O(n)$, but it is higher than for GD since CG computes additionally momentum term. SR-1 computation complexity is $O(n^2)$, because of outer vector product in rank-one correction and further matrix-vector product. So the final sorting is 1) GD, 2) CG, 3) SR-1, 4) Newton.

6. (LCQ) All inequality and equality constraints are given by non-degenerate affine functions, i.e. $g_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} + b_i$, where $\boldsymbol{a}_i \neq \boldsymbol{0}$.

   (LICQ) A set of vectors $\{\nabla g_i(\boldsymbol{x}), \nabla h_j(\boldsymbol{x}) \mid (i,j) \in \text{Active}(\boldsymbol{x})\}$ are linear independent

   (Slater) For convex constrained optimization problem there exists a strict interior point, i.e. $\exists \tilde{\boldsymbol{x}} : g_i(\tilde{\boldsymbol{x}}) < 0 \ \forall i, \ h_j(\tilde{\boldsymbol{x}}) = 0 \ \forall j$.

   Example of non-regular optimization problem:

$$\begin{aligned} &x \to \min_x, \\ &x^2 \leq 0. \end{aligned}$$

7. Let's write down Lagrange function:

$$L(\boldsymbol{x}, \mu) = \boldsymbol{c}^T \boldsymbol{x} + \sum_i x_i \log(x_i) + \mu(\sum_i x_i - 1).$$

   Let's differentiate this function w.r.t. x_i:

$$\frac{\partial L}{\partial x_i} = c_i + \log(x_i) + 1 + \mu = 0 \ \Rightarrow \ x_{opt,i} = \exp(-c_i - 1 - \mu).$$

   Substitute this result into the constraint:

$$1 = \sum_i x_i = \sum_i \exp(-c_i - 1 - \mu) = \exp(-1 - \mu) \sum_i \exp(-c_i) \ \Rightarrow \ \exp(-1 - \mu) = \frac{1}{\sum_i \exp(-c_i)}.$$

   Finally,

$$x_{opt,i} = \frac{\exp(-c_i)}{\sum_{j=1}^n \exp(-c_j)}.$$

   Formally, we need to solve the optimization problem also w.r.t. the constraint $x_i > 0$. Our found solution $\exp(-c_i)/\sum_j \exp(-c_j)$ satisfies this constraint, so this is indeed the answer.

8. First let's write down Lagrange function:

$$L(\boldsymbol{x}, \lambda) = \boldsymbol{c}^T \boldsymbol{x} + \lambda(\|\boldsymbol{x}\|_2^2 - b)$$

   Dual function by definition is:

$$D(\lambda) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda).$$

   So let's differentiate Lagrange function w.r.t. $\boldsymbol{x}$:

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda) = \boldsymbol{c} + 2\lambda \boldsymbol{x} = 0 \ \Rightarrow \ \boldsymbol{x}_{min} = -\frac{1}{2\lambda}\boldsymbol{c}.$$

   Substituting this result back to the Lagrange function we find:

$$D(\lambda) = L(\boldsymbol{x}_{min}, \lambda) = -\frac{\|\boldsymbol{c}\|^2}{2\lambda} + \lambda \frac{\|\boldsymbol{c}\|^2}{4\lambda^2} - \lambda b = -\frac{\|\boldsymbol{c}\|^2}{4\lambda} - \lambda b.$$

   Hence, dual optimization problem would be the following:

$$\begin{aligned} &-\frac{\|\boldsymbol{c}\|^2}{4\lambda} - \lambda b \to \max_\lambda, \\ &\lambda \geq 0. \end{aligned}$$

9. Let's use Epigraph transformation:

$$\max_i(\boldsymbol{a}_i^T \boldsymbol{x} - b_i) \to \min_{\boldsymbol{x}} \ \Leftrightarrow \ \begin{cases} t \to \min_{t,\boldsymbol{x}}, \\ \max_i(\boldsymbol{a}_i^T \boldsymbol{x} - b_i) \leq t \end{cases} \quad \Leftrightarrow \ \begin{cases} t \to \min_{t,\boldsymbol{x}}, \\ \boldsymbol{a}_i^T \boldsymbol{x} - b_i \leq t \ \forall i \end{cases}$$

   The last problem is a linear programming problem since all functions and constraints are linear.

10. First let's prove that the function $f(\boldsymbol{x}) = \sum_{i<j} |x_i - x_j|$ is convex. The one-dimensional function $|y|$ is convex, the function $|x_i - x_j|$ is a superposition of convex function with affine transformation of argument and thus is convex, and finally the function $f(\boldsymbol{x})$ is a sum of convex functions with positive coefficients, so it is convex.

Let's rewrite the function $|x_i - x_j|$ as a composition with affine transformation:

$$|x_i - x_j| = |\boldsymbol{a}_{ij}^T \boldsymbol{x}|, \; \boldsymbol{a}_{ij}^T = [0, \dots, 0, \underbrace{1}_{\text{i-th pos}}, 0, \dots, 0, \underbrace{-1}_{\text{j-th pos}}, 0, \dots, 0], \; \boldsymbol{a}_{ij} \in \mathbb{R}^n$$

Then

$$\partial f(\boldsymbol{x}) = \partial \sum_{i<j} |x_i - x_j| = \sum_{i<j} \partial |x_i - x_j| = \sum_{i<j} \partial |\boldsymbol{a}_{ij}^T \boldsymbol{x}| = \sum_{i<j} \boldsymbol{a}_{ij} \partial |\cdot| (\boldsymbol{a}_{ij}^T \boldsymbol{x})$$

Here we use the general rule for finding subdifferential for affine transformation.

Also let's check the correct dimension of the output. Subdifferential $\partial f(\boldsymbol{x})$ should consist of vectors from $\mathbb{R}^n$. $\partial |\cdot|$ is one-dimensional object, multiplication by $\boldsymbol{a}_{ij}$ gives us n-dimensional object.

11. We need to find

$$f^*(s) = \sup_{x>0} \left(xs - \frac{1}{x} \right).$$

Let's take derivative of the function $xs - 1/x$ w.r.t. x:

$$s + \frac{1}{x^2} = 0 \;\Rightarrow\; x^2 = \frac{1}{-s}.$$

This is possible only if $s < 0$. Then $x_{opt} = 1/\sqrt{-s}$. This is indeed maximum because the second derivative $-2/x^3$ is negative for all $x > 0$ (so the function is strictly concave). Then $f^*(s) = s/\sqrt{-s} - \sqrt{-s} = -2\sqrt{-s}$.

Also we need to consider other options for s. If $s = 0$, then $\sup_{x>0}(xs - 1/x) = \sup_{x>0}(-1/x) = 0$. If $s > 0$, then $\sup = +\infty$ and conjugate function is not defined. So finally

$$f^*(s) = -2\sqrt{-s}, \text{ if } s \leq 0.$$

12. Let's consider composite minimization problem:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + h(\boldsymbol{x}) \to \min_{\boldsymbol{x}}.$$

Here $f(\boldsymbol{x})$ is convex and smooth function, $h(\boldsymbol{x})$ is convex and closed function.

General scheme of Prox-GD:

- Initialize $\boldsymbol{x}_0$;
- For $k = 0, 1, 2, \dots$:
  - $\boldsymbol{x}_{k+1} = \text{prox}_{\alpha h}(\boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k))$.
  - Compute gradient mapping: $G(\boldsymbol{x}_k) = \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k+1}}{\alpha}$. If $\|G(\boldsymbol{x}_k)\|_2^2 \leq \varepsilon$, then stop.

Here $\text{prox}_h(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \left(\frac{1}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + h(\boldsymbol{y}) \right)$. The Prox-GD is a descent optimization method, i.e. $F(\boldsymbol{x}_{k+1}) < F(\boldsymbol{x}_k)$ for all reasonable step sizes.